Deployment in dynamic environments

José L. Ruiz
Departamento de Ing. Telemática
Universidad Politécnica de Madrid

Juan C. Dueñas(DIT), Fernando Usero(TELVENT), Cristina Díaz(Telefonica I+D)

Index

- Introduction
- Deployment in service oriented architectures
- Deployment requirements
- Resolvit:
  - Dependency model
  - Deployment algorithm
- Related work
- Conclusions and future work

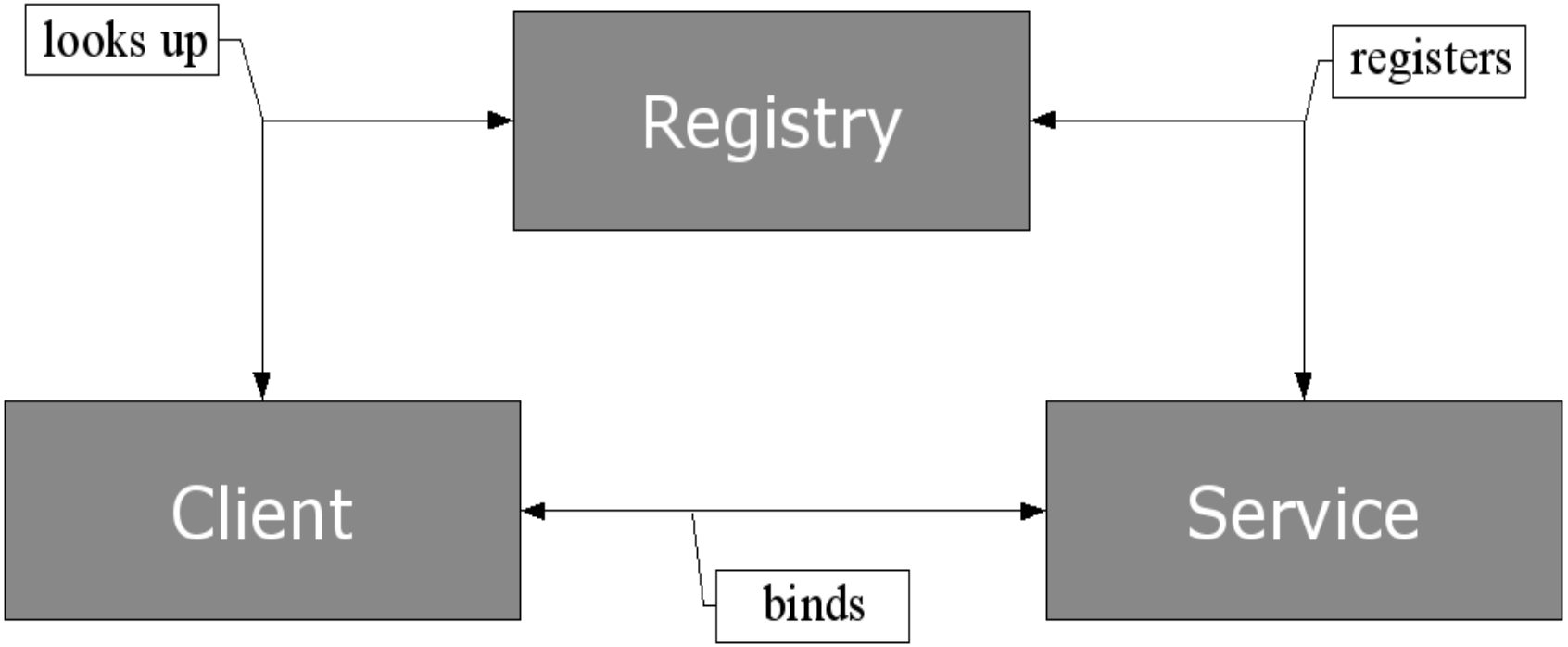
Introduction

- Deployment is becoming a key point:
  - Evolution of consumer electronics
  - Economies of scale are reducing costs
  - Internet, the global market
  - Shorter development and release cycles
- Service oriented architectures
  - Dynamicity is the key: discover, bind and use
  - Cooperative approach:
    - Increase added value
    - … but also a way to survive in an aggressive environment

Index

- Introduction
- Deployment in service oriented architectures
- Deployment requirements
- Resolvit:
  - Dependency model
  - Deployment algorithm
- Related work
- Conclusions and future work

Service Oriented Architectures



OSGi Open Services Gateway initiative

- Service Oriented Java-based architecture
- Embedded systems is its main target
- Well identified domains:
  - Home gateway
  - Mobile
- New domains are appearing thanks to:
  - Dynamically install and update software
  - Class loading capabilities
- Just some details:
  - Execution environment(service framework)+ Services
  - Deployment units are bundles (JAR files)
  - Provides a registry for services, which are a Java interface
  - Resolves Java packages dependencies among installed bundles

Deployment in SOA

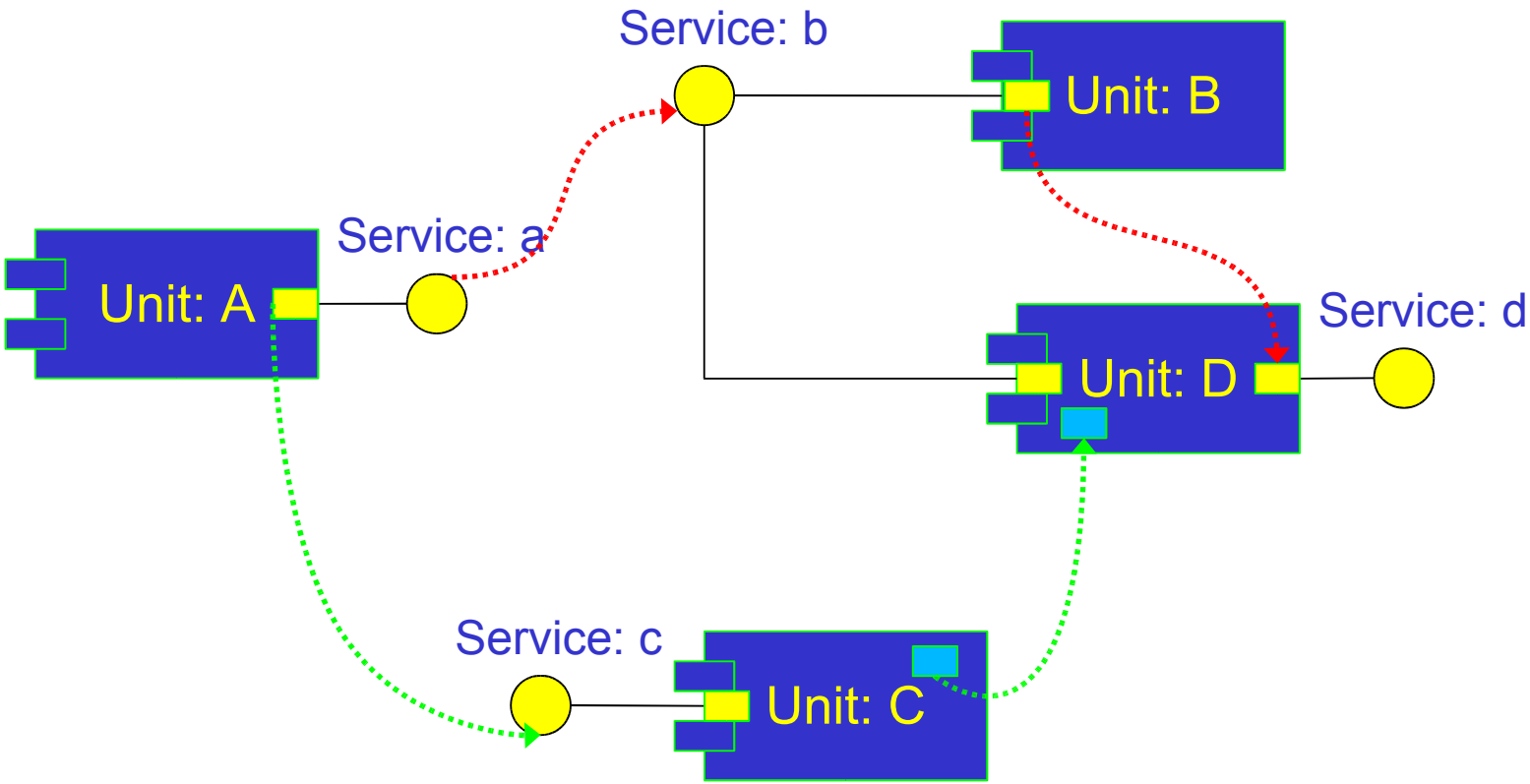
- Applications are built at runtime by collaboration among services
- Services can be dynamically: bound, deployed, reconfigured, uninstalled and updated
- Our aim:

"To develop an automated deployment engine for dynamic envirnoments that operates without human intervention"

Dependencies in SOA

- Service definition: may be understood as any piece of software that offers a certain functionality, specified by means of an interface
- Dependencies, two types:
  - Visible in the service interface
  - Invisible in the service inter
- We have to deal with both!!

Understanding dependencies



Resolving dependencies

- Do not forget it is not just a functional/code aspect
- To deal with invisible dependencies:
  - Instrospection
  - Metadata files
- Metadata files:
  - Clear separation between deployer and developer
  - Packaged with the deployment unit and available separately:
    - Dependencies' analysis without unit download
    - Optimize through caching

Index

- Introduction
- Deployment in service oriented architectures
- Deployment requirements
- Resolvit:
  - Dependency model
  - Deployment algorithm
- Related work
- Conclusions and future work

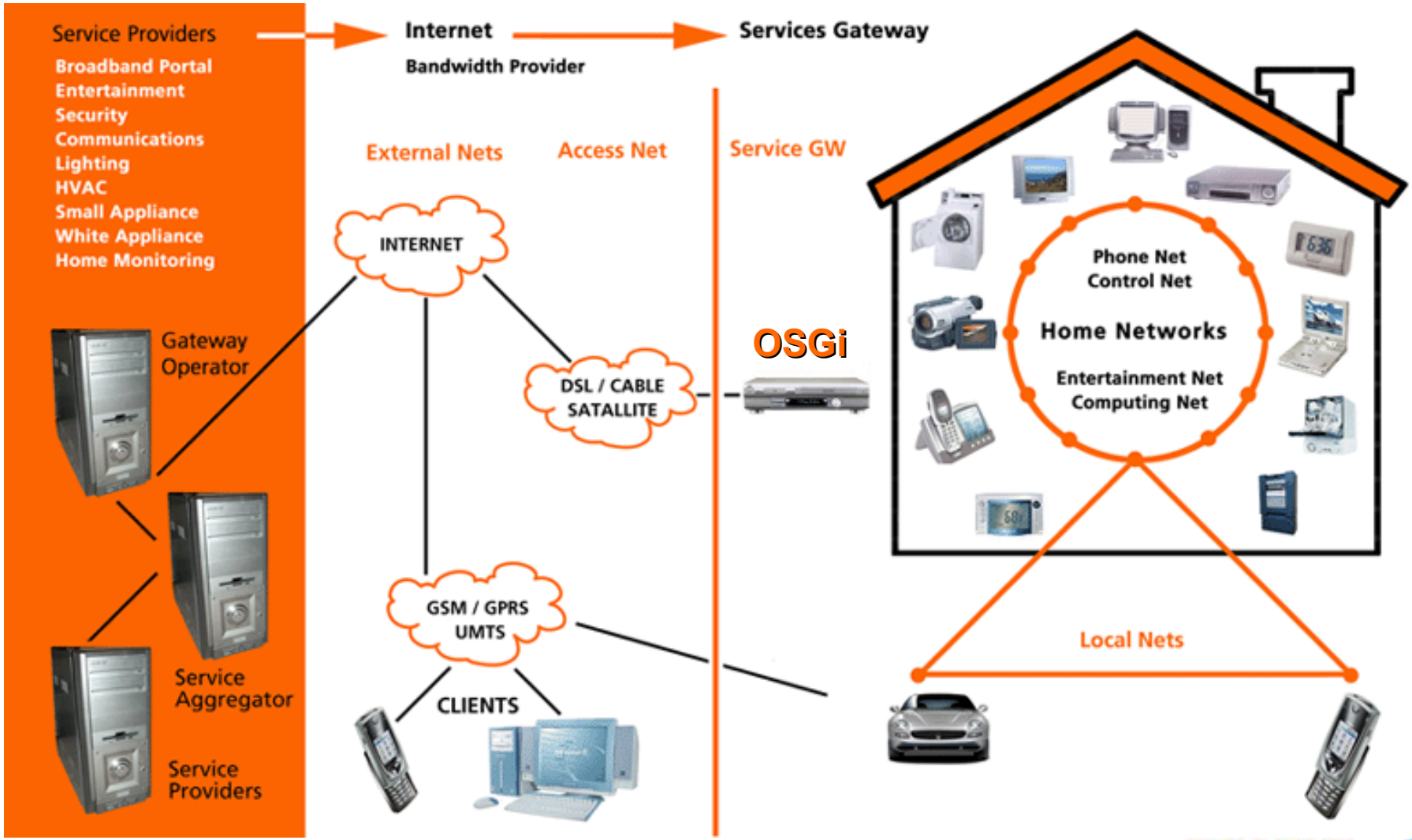
Deployment requirements

- The **status** of the platform must be stored
- It must be **complete** and therefore:
  - We need means to express dependencies
  - Deployment must span **multiple layers** of the platform
- Software **evolution** must be supported:
  - Version aware deployment
- It must be **safe** and **reliable**
- Deployment process should be **distributabe**
- It should be **context and cost aware**
- It must support:
  - **Multi-vendor** provisioning

Index

- Introduction
- Deployment in service oriented architectures
- Deployment requirements
- Resolvit:
  - Dependency model
  - Deployment algorithm
- Related work
- Conclusions and future work

End 2 end provisioning

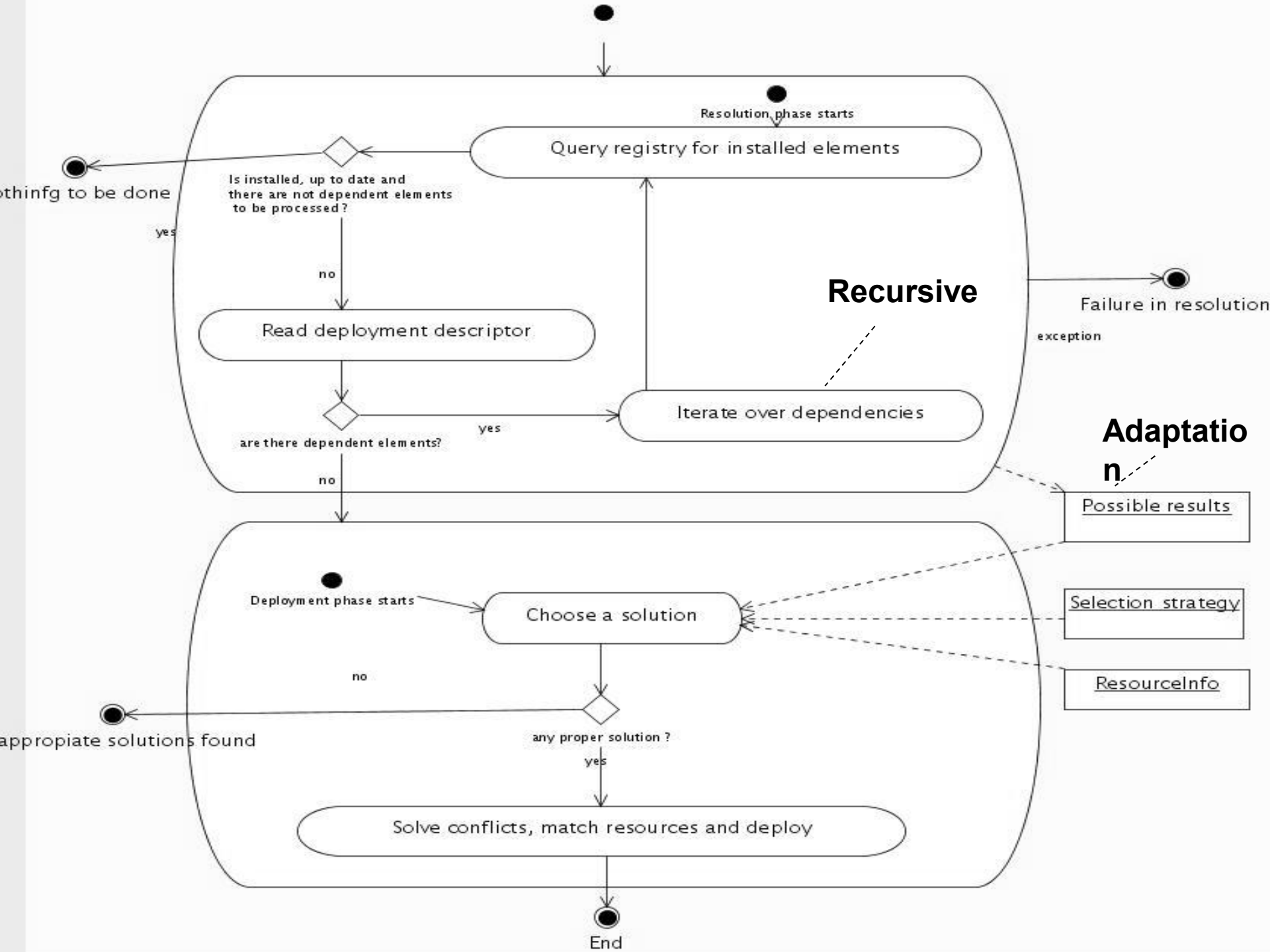


Resolvit

- Deployment engine for OSGI enabled devices
- Features:
  - Dependency model driven resolver
  - Multilayer deployer: native and OSGi
  - XML metadata specified by an XML Schema
  - Metadata files are packaged with the bundles but also available separately

Dependency Model

- Dependencies are defined by means of Sources and EndPoints:
  - Source, identifies a deployment unit
  - EndPoint identifies either a deployment unit or something contained inside one
- Boole's algebra operators:
  - AND, this type will be used to link a source with endpoint elements that are mandatory for the source's correct execution. Endpoints will have to be deployed before the source on the platform.
  - NOT, whenever an identified conflict between a source and a certain endpoint is known. If the endpoint resource is already deployed on the platform, then the deployment engine will have to take a decision, because both elements cannot live together.
  - OR, this type will be used when one or more resources can be selected at the same time.
  - XOR, this type will be used when either the source or the endpoint may be chosen, but not at the same time.


Resolution phase starts
Query registry for installed elements
Is installed, up to date and there are not dependent elements to be processed?
othinfg to be done
yes
no
Read deployment descriptor
are there dependent elements?
yes
Iterate over dependencies
Recursive
Failure in resolution
exception
Adaptation
Possible results
Selection strategy
ResourceInfo
Deployment phase starts
Choose a solution
no
appropiate solutions found
any proper solution ?
yes
Solve conflicts, match resources and deploy
End

Index

- Introduction
- Deployment in service oriented architectures
- Deployment requirements
- Resolvit:
  - Dependency model
  - Deployment algorithm
- Related work
- Conclusions and future work

Related Work

- Operative systems:
  - Windows update
  - Debian GNU/Linux
- OMG Deployment and configuration for component oriented distributed applications
- OBR Oscar Bundle Repository

Conclusions

- Overview of deployment issues in dynamic enviroments:
  - Dependencies:
    - Visible and invisible
    - Not only a functional aspect
  - Identified requirements
- Resolvit:
  - Deployment engine for OSGi enabled devices
  - Dependency model
  - Deployment algorithm

Future work

- Define the architecture for a more intelligent repository
- Implement policies:
  - QoS aware
  - Context aware
  - Oriented to the user

Thanks

Jose L Ruiz, jlruiz@dit.upm.es
Telemática, ETSI Telecomunicación
Ciudad Universitaria, sn, Madrid, E-28040 Spain