# When Applications can Roam Freely
## OSGi Service Platform R4

Peter Kriens

*Technical Director OSGi*

*Peter.Kriens@osgi.org*

# Contents

- Why the OSGi Service Platform
- The Software Problem
- Service Architectures
- The OSGi Service Platform
- Conclusion

# Why the OSGi Service Platform?

- What problems does the OSGi Service Platform address?
- A unified software market:
  - The limited (binary) software portability problem
  - The complexity of building heterogeneous software systems
    - Supporting the myriad of configuration, variations, and customizations required by today's devices
  - Managing the software life-cycle on the device

# Limited Binary Software Portability

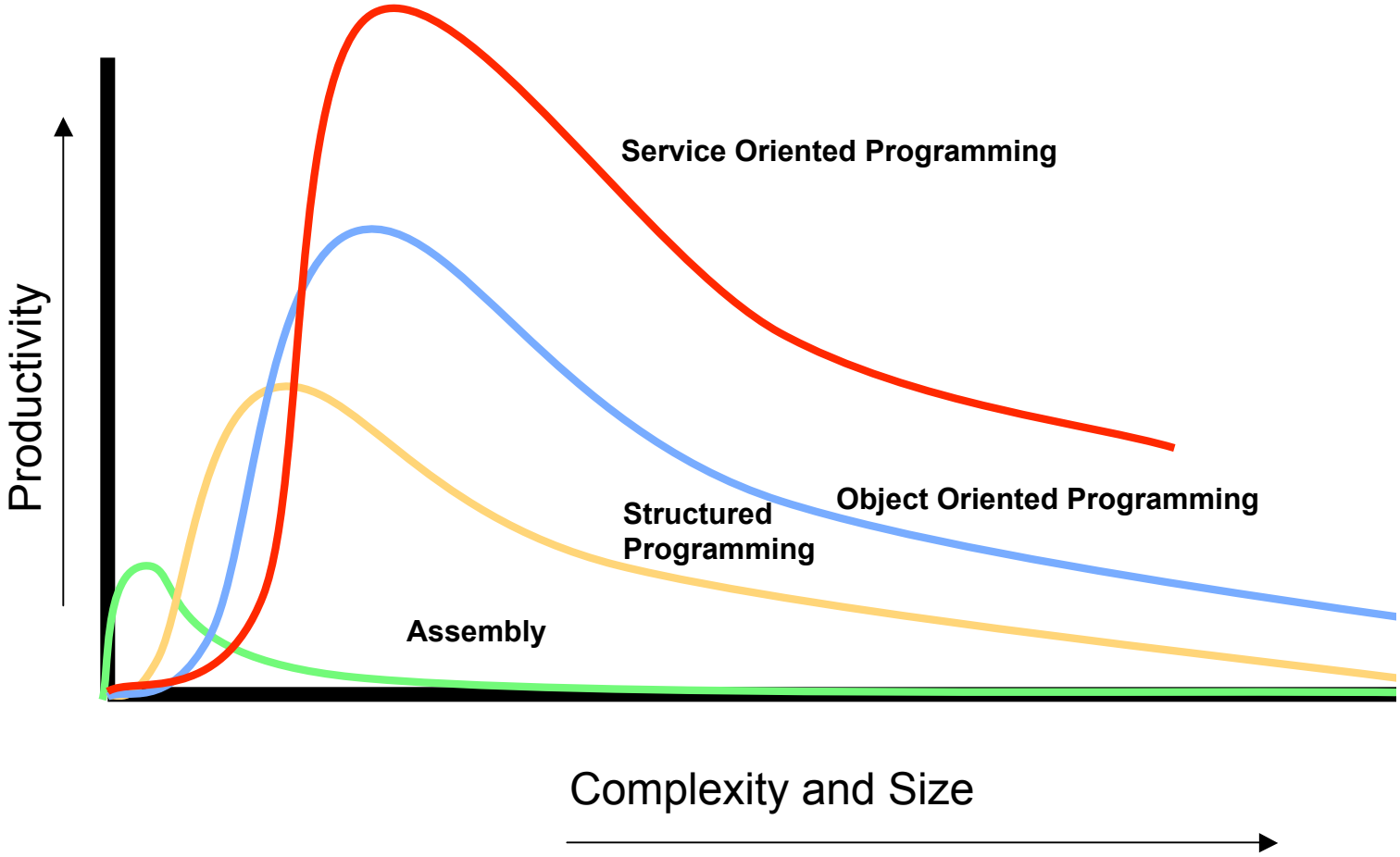
- Lack of portability causes
  - Market friction: No large market of reusable components and applications
  - Reduced quality
- Unnecessary constraints on hardware and software architectures
  - CPUs differ widely in cost and performance
  - Linux is nice, but it is sub-optimal for smaller devices
- Benefits of the OSGi Platform
  - Applications run unmodified on different hardware and software architectures

# Complexity of Software

- A DVD player can contain 1 Million lines of code
  - Comparison: Space Shuttle ~ 0.5 Million
- A BMW car can contain up to 50 networked computerized devices
- Eclipse contains 2.5 million lines of code
- An average programmer writes an average of 10 lines a day …

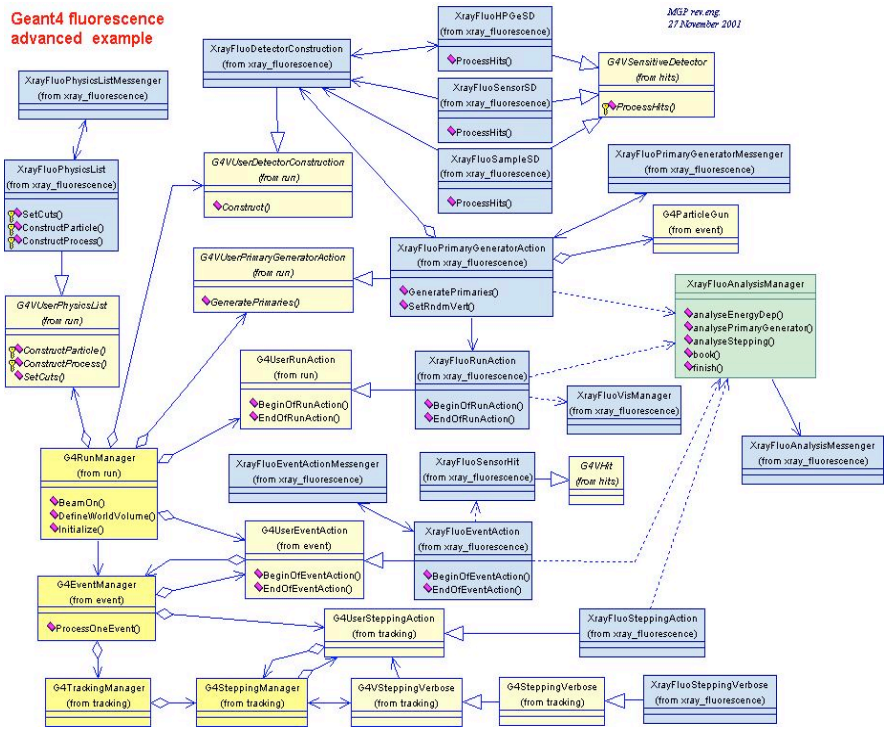
- Houston … we have a problem

# Complexity of Software



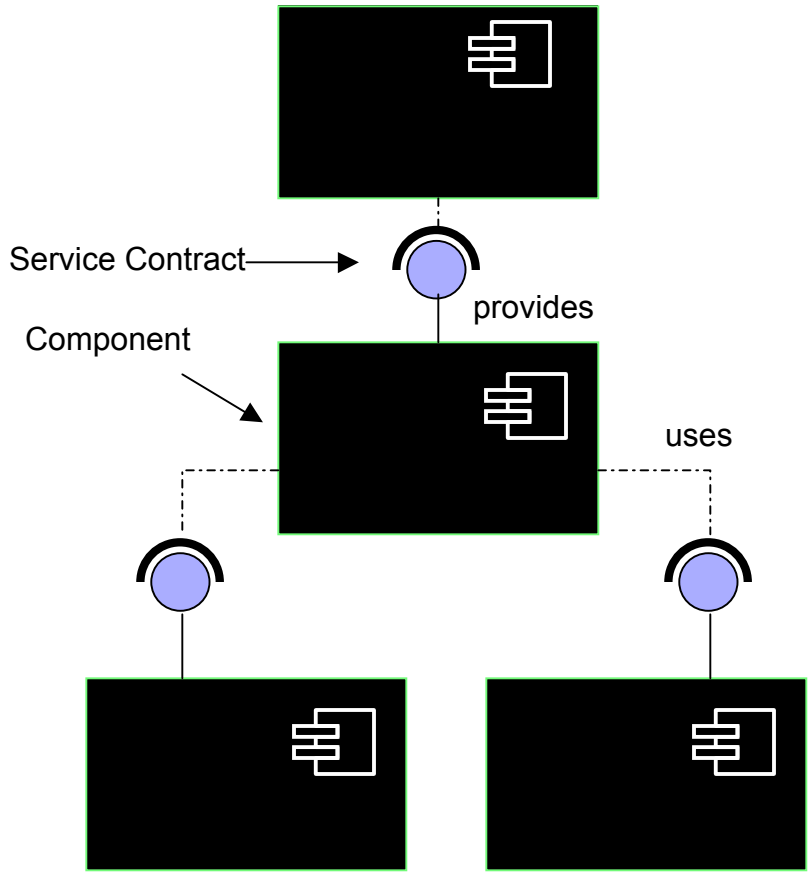
# Limits Object Oriented Technology

- Objects are great, but oh, the tangled webs we weaves …
- Coupling severely limits reusability
  - Using a generic object, can drag in a large number of other objects
- Creates overly large systems after a certain complexity is reached
- Flexibility must be built in by the programmer
  - Plugin architectures



# Service Oriented Architectures

- Separate the contract from the implementation
  - Allows alternate implementations
- Dynamically discover and bind available implementations
  - Based on contract (interface)
- Components are reusable
  - Not coupled to implementation details



# OSGi Service Registry

- Provides an in-VM service model
  - Discover (and get notified about) services based on their interface or properties
  - Bind to one or more services by
    - program control,
    - default rules, or
    - deployment configuration
- The OSGi Alliance provides many standardized services
- SOA Confusion
  - Web services bind and discover over the net
  - The OSGi Service Platform binds and discovers inside a Java VM

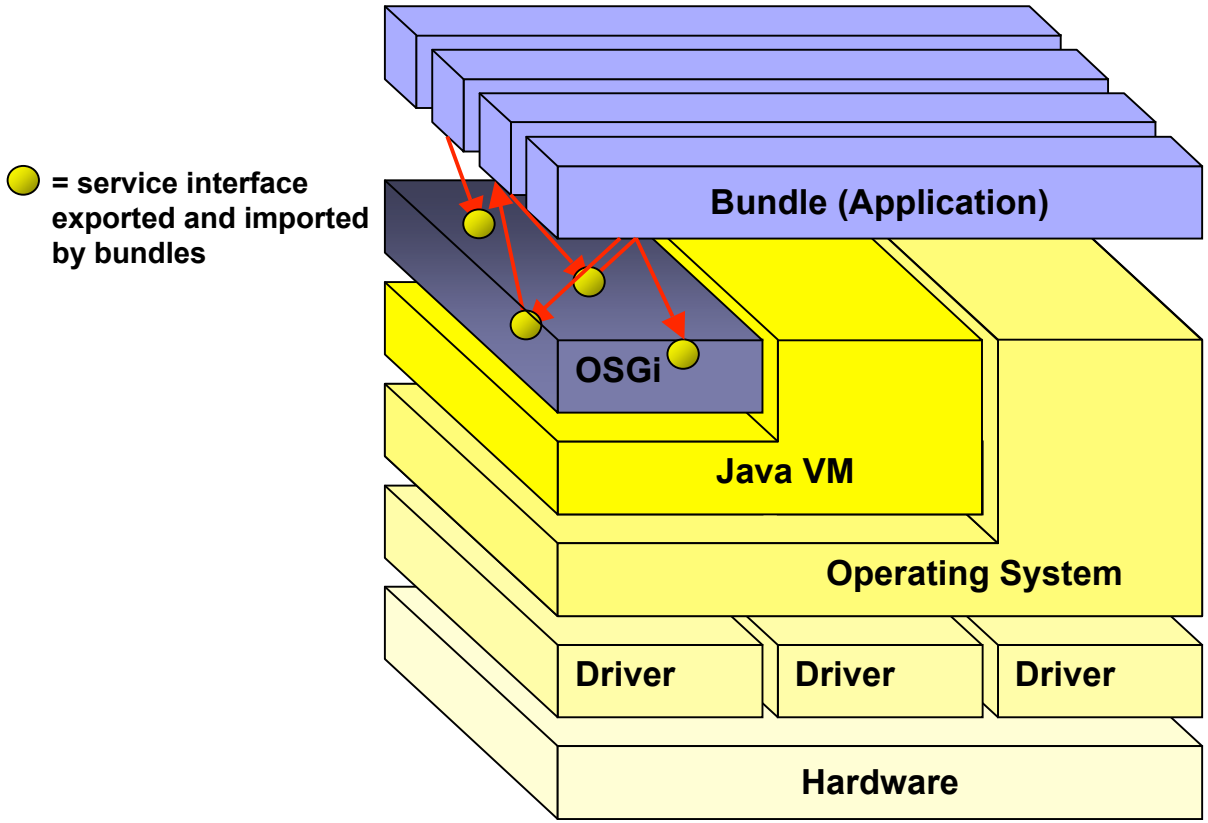
- OSGi Service Platform Benefits:
  - Components are smaller (easier to make) and not coupled to other components (gives reusability)
  - Excellent model for the myriad of customizations and variation that are required of today's devices
  - Collaboration model

# Device Management

- The software life-cycle does **not** stop when a networked device leaves the factory
- Updates and new installs are a fact of life
- (Remote) Management is an intrinsic and non-trivial aspect of today's device software
- The OSGi Alliance has standardized the API for remote device management

- Benefits:
  - Supports any number of management protocols
  - Optimized solutions for specific problems
  - Reduces management costs

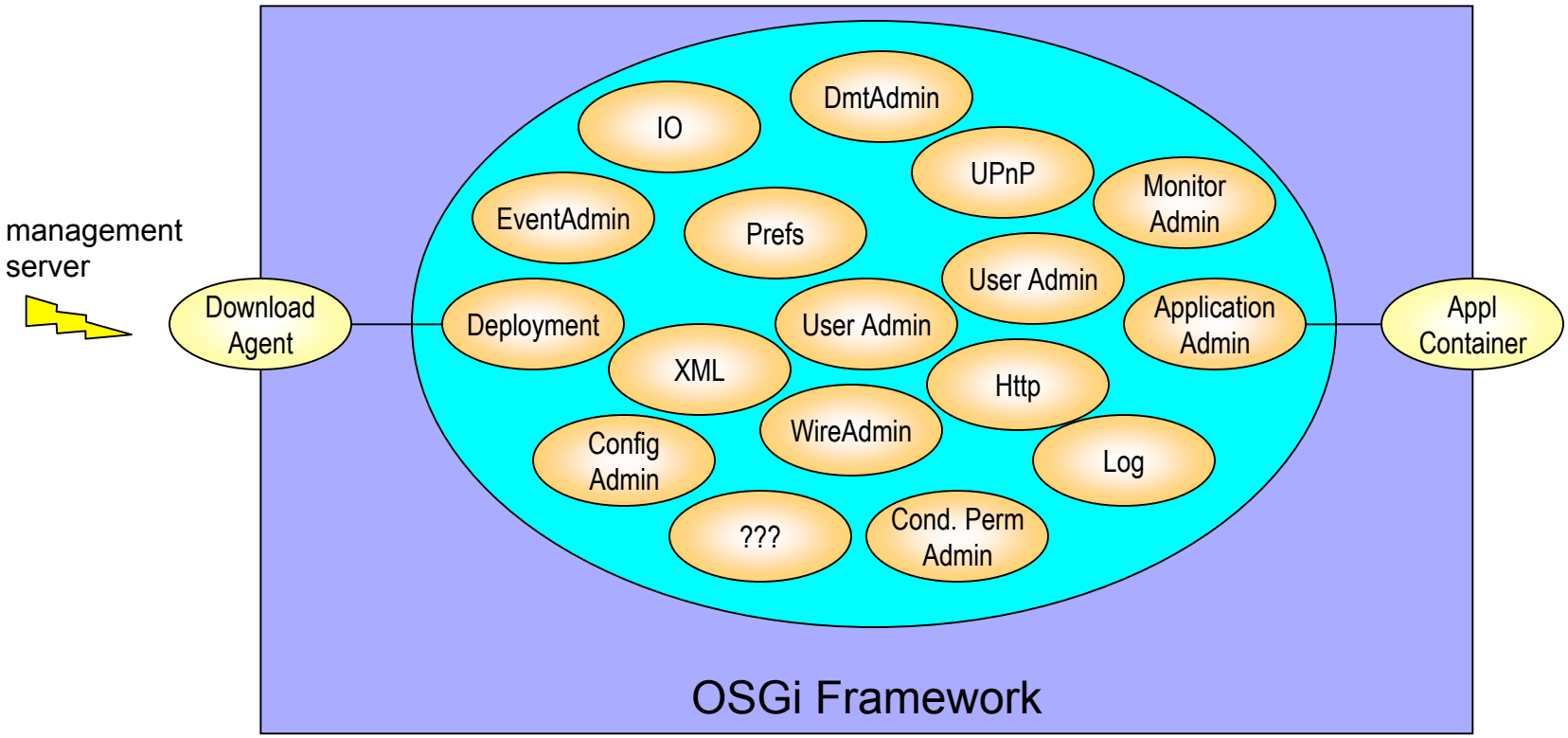
# OSGi Environment



# Framework

- Allows applications to share a single Java VM
  - Isolation/Security
  - Communication between applications
  - Collaboration between applications
  - Life cycle management
- Policy free
  - Policies are provided by *bundles*
  - API is fully self managed

# Overview OSGi Service Platform



# Conclusion

- The OSGi Service Platform provides an excellent environment for system, firmware, middleware and application software
- The service architecture solves many of the complex customization issues that are part of massive market devices like mobile phones and telematic units.
- The security model is the most fine grained model available without becoming unmanageable