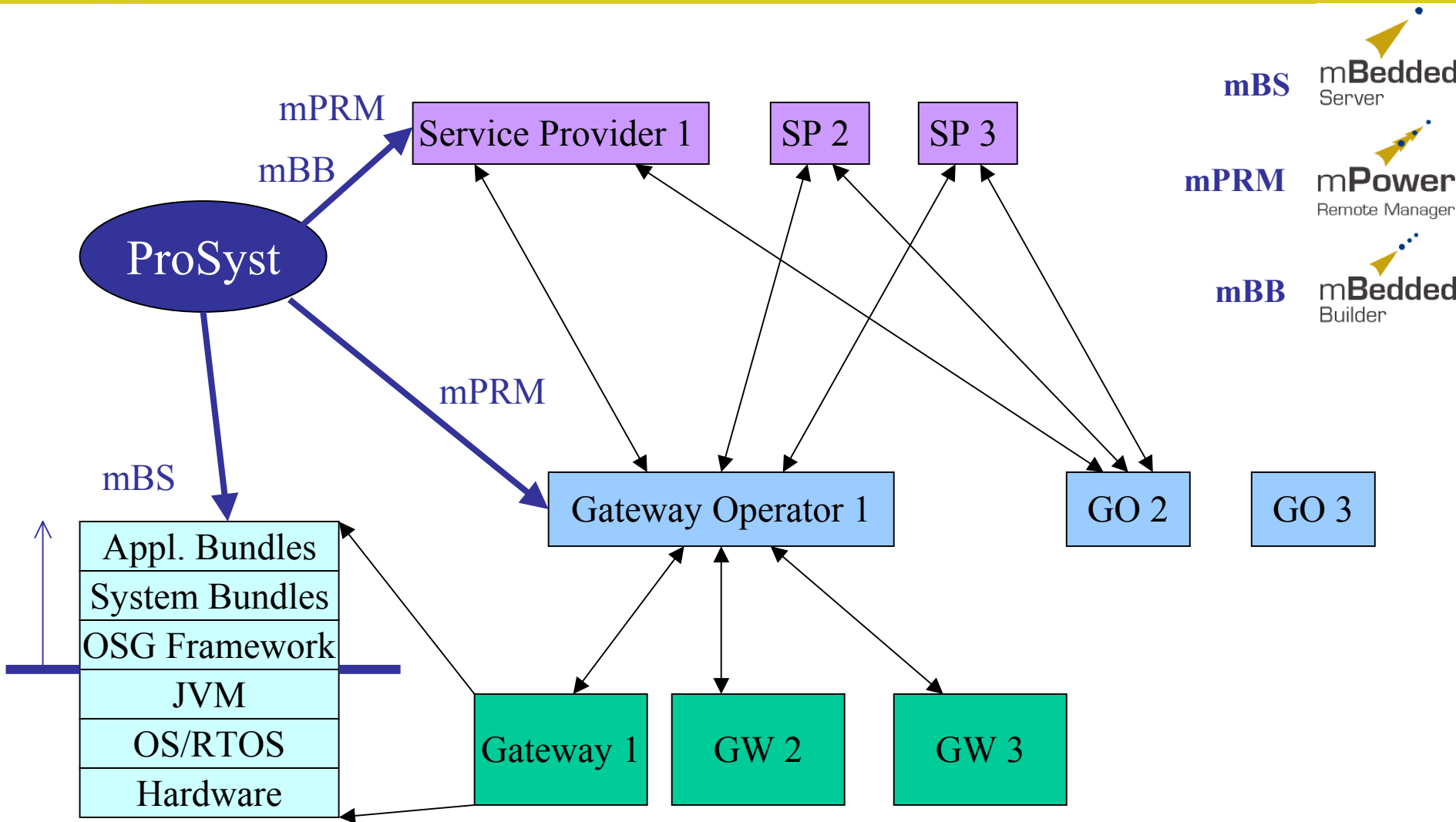


*Available OSGi service platforms*

**What distinguishes ProSyst's offering?**

Dr. Dimitar Valtchev,  
CTO

- **Product structure**
- **Service gateway platform (mBS)**
- **Management platform (mPRM )**
- **Conclusions**



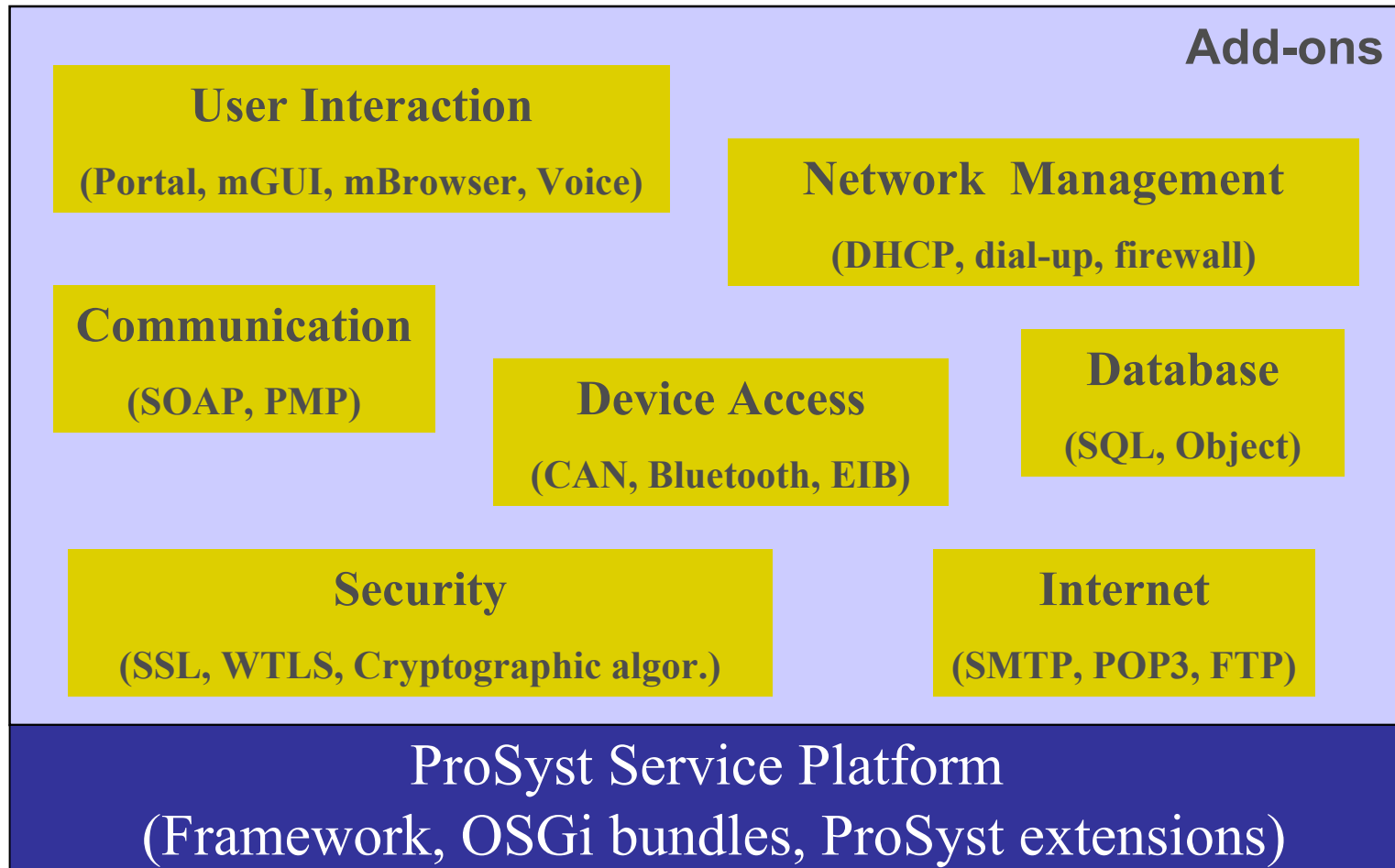


**mBedded**  
Server

**Fully compliant with  
OSGi Release 2**

- **Modular architecture - subsystems with clear interfaces so every subsystem can be easily replaced (security, class loading, resource management, etc.)**
- **Framework configuration**
- **Resource management**
  - ➔ Memory control (object pooling)
  - ➔ Thread control and optimizations (thread pooling)
  - ➔ Storage control
  - ➔ CPU time control
- **Power monitoring and management**
- **Minimized dependency on the Java VM class library in order to support more VMs**

- **Flexible file system**
  - Support for different kinds of persistent storage
  - Support for gateways without persistent storage
  - Independence from the JVM java.io package
  - Control of how much space is used by each bundle
- **Bootstrapping and framework initialization**
- **Minimized framework (as much functionality as possible in bundles)**
- **Minimized package dependency between bundles. The extensively used packages are in separate bundles**



- **Configurator tool** - building of server images. The images include VM, framework implementation, a ready pre-built storage, scripts for a particular VM and OS, deployment of the image to the board.
- **Verification tool** - configuration of bundles, checking if this configuration can be resolved (internal links between the bundles or if this configuration will work on a particular execution environment).
- **Tool for bundle packaging and editing**
- **Tools for protocol simulation, device control and monitoring**

- **PowerPC**
- **StrongARM and Arm**
- **SH 4**
- **MIPS**
- **XScale**
- **x86**

- **Desktop OS**
  - Linux
  - Solaris
  - Windows
- **Embedded OS and RTOS**
  - Linux (Hard Hat, Embedix, etc.)
  - VxWorks
  - QNX Neutrino
  - Windows CE, Windows CE Automotive
  - Jbed (Esmertec)
  - Intent (Tao)

### General requirements

- Compatible with JVM specification
- Required java packages: java.lang, java.lang.reflect, java.util, java.util.zip, java.math, java.io, java.net, java.security. Some of these packages are optional
- JNI support, dynamic class loading support

## Virtual machines list

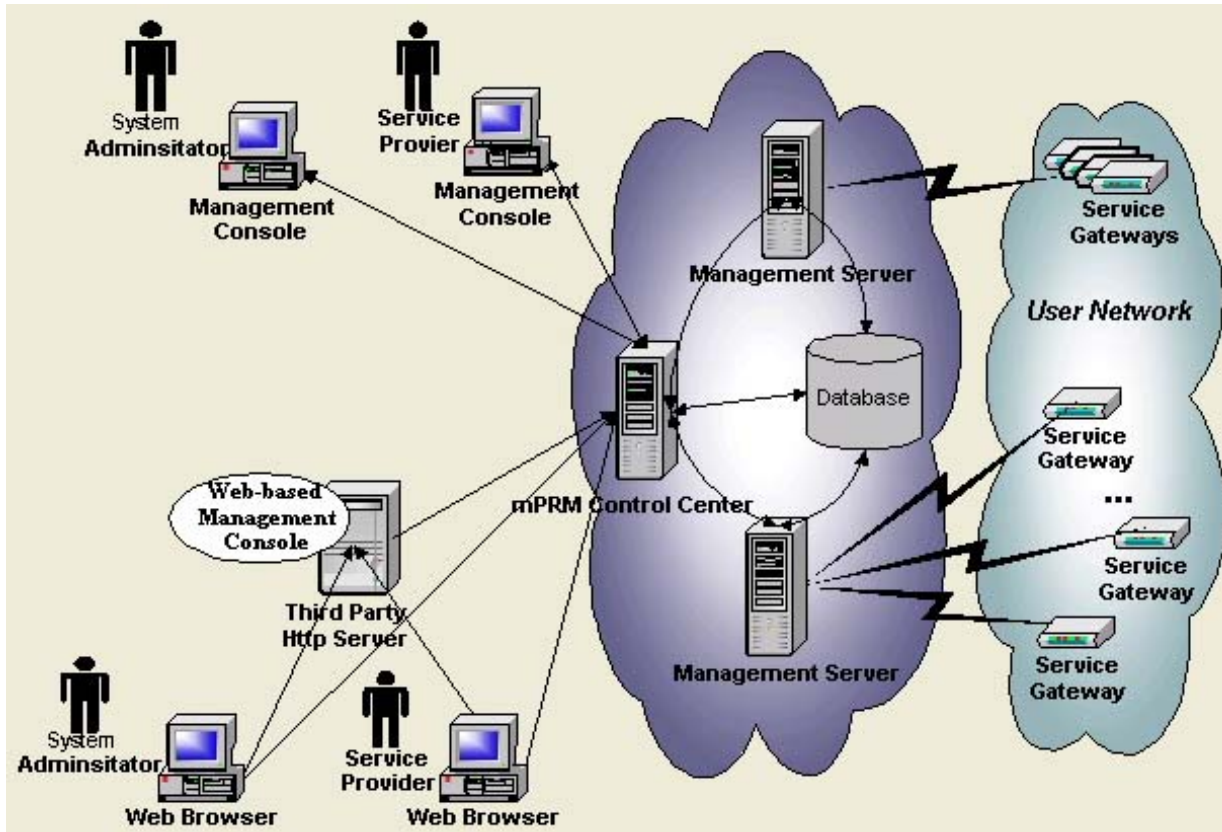
- SUN and IBM JDK 1.1.7 / 1.1.8 / 1.2.2 / 1.3 / 1.4
- Personal Java version 3.1
- OTI J9
- Insignia Jeode
- HPChai
- JSCP
- PERC
- CVM (Foundation Profile)
- Kaffe
- TAO Intent
- Jbed
- Skelmir
- SavaJe



- Supports Remote Management Architecture defined in OSGi RFC 20
- Initial provisioning complies with OSGi RFC 27 & 36
- Management CIM Model RFC 19
- Accounting service complies with OSGi RFC 14
- User management complies with OSGi RFC 10
- Preferences service complies with OSGi RFC 09
- Permission Admin RFC 17, Configuration Management RFC 7, Metatyping API RFC 16, Log Service RFC 3

- **Gateway Registration and Initial Provisioning**
- **Network Communication >**
- **User Management**
- **Subscription Management and Service Delivery >**
- **Billing**
- **Integration >**
- **User Interaction**
- **Application Platform >**

## System Architecture



- Control Center
- Management Server(s)
- Database
- Service Gateways
- Web Server (optional)
- Management Console(s)

## Network Communication

- Provides messaging-based communication mechanism between service gateways and backend management system
- Manages communication sessions for connected gateways
- Communication is not bound to any specific network transport. Backend servers are able to communicate over different transports with different service gateways.
  - ➔ UDP-based communication, with extension to ensure reliability and consistency of communication.
  - ➔ Http-based communication
  - ➔ Provision of API and support for development and deployment of other protocols
- Ensures security and integrity of communication. SSL-based encryption and authentication.

## Subscription Management and Service Delivery

- Provides user-oriented subscription management
- Provides support for different service delivery policies
  - ➔ Automatic - bundle(s) are installed at time of subscription on all gateways belonging to the subscribed user and are ready to use
  - ➔ On demand - bundles are installed when the user activates the service on a particular gateway and can be uninstalled when the user finishes using the service
- Provides mechanism for different charging models:
  - ➔ Initial subscription price; time period base fee; service defined
  - ➔ Charge once per user; charge per every gateway
- System ensures that all bundles supporting a given service will be available on the gateways, according to delivery policies

## Integration

- Allows integration with J2EE-compatible application servers
  - ➔ Provides JCA-compatible module which ensures mPRM connectivity and access to resources managed by mPRM
  - ➔ Provides JMS bridge between services running on service gateways and J2EE applications
- Uses WBEM (CIM) interface to resources managed by mPRM

## Application Platform

- Provides support for installing custom modules on the backend side
- Provides service APIs to mPRM functions, which can be used by these custom modules
- Provides remote access API to mPRM functions for integration with external system and applications
- Provides API on the gateway which can be used by custom services to access mPRM functionality and/or other custom back-end modules

- OSGi specification completely defines a powerful service delivery platform
- At the same time, it leaves a lot of freedom for OSGi vendors in dealing with some architectural and implementation issues
- Depending on how this freedom is used, OSGi implementations of differing quality co-exist (portability, stability, resource usage, error detection, fail-recovery, etc.)
- There are significant differences in the convenience of working with the frameworks (configuration tool, verification tool, device browsers, etc.)

- Some of the vendor-specific features do not need to be specified by OSGi (e.g. management protocol, GUI frameworks)
- Many others, however, must be specified in order to avoid redundancy and to use the benefits of the OSGi approach fully
  - ➔ Specifying standard services like thread pool, object pool, timer, etc.
  - ➔ Abstraction for working with devices independently of the underlying protocol
  - ➔ Defining (or adopting) Java interfaces for some important protocols in residential and automotive areas
  - ➔ Adopting suitable APIs from other organizations like JCP, AMI-C, JConsortium, etc.
  - ➔ Defining standard headers for some resource requirements (memory, storage, CPU)
  - ➔ Multiple frameworks, ...

## ProSyst advantages

- Complete software solution (gateway, back-end, development tools)
- Many ready-to-use bundles packages
- mBS runs on all important JVMs
- Customizable framework
- Close integration with several JVMs
- All ProSyst bundles are tested on the OSGi reference implementation, IBM SMF, and Gatespace framework
- Reliable and realistic OSGi management system today
- Easy-to-use products, detailed documentation, numerous demos
- Integrated environment for OSGi development, some of the plug-ins are ported for JBuilder and Eclipse

## Any questions?

- 1. Ask me in the next minutes**
- 2. Contact us during the Congress**

Dimitar Valtchev - CTO

Hristo Stanchev - Lead Architect

Thomas Hott - CEO

Susan Schwarze – Marketing Director

Kai Hackbarth – Product Manager