*Draft*

# Regions in Virgo (v0.1)

Glyn Normington

January 7, 2011

The aim is to model how Virgo divides the OSGi framework into a connected graph of *regions*. The model will inform the work on bug 330776 “Re-implement user region using framework hooks instead of nested framework”.

**The model is incomplete. It does not cope with importing from more than one region to a given region because the “squashed” result may not be a region.**

## 1 Introduction

The Virgo kernel is isolated from applications by the use of *regions*. The kernel runs in its own region and applications run in a *user region*.

Virgo 2.1.0 implemented the user region as a nested framework, but Equinox has deprecated the nested framework support in favour of the framework hooks which are being defined for OSGi 4.3. Bug 330776 re-implements the user region using the OSGi framework and service registry hooks.

Framework hooks are used to limit which bundles can 'see' particular bundles and exported packages, and service registry hooks are used to limit which bundles can 'see' particular services. 'Seeing' includes both finding and being notified via lifecycle events.

Rather than allowing arbitrary hook behaviour, we limit the hooks to operate on regions which are connected together with filters.

A *region* is then a set of bundles and a region can see bundles, exported packages, and services from another region via a *connection*. Each connection has a filter which may limit what can be seen across the connection. Hence regions and connections form a directed graph decorated by filters.

For example, Figure 1 shows three regions connected by three connections. Each connection has a filter which limits what bundles, exported packages, and services are visible across the connection.

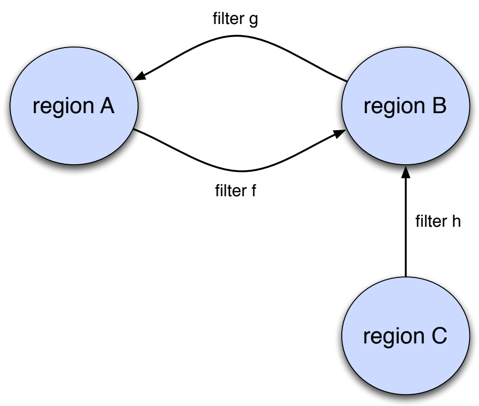

Figure 1: Connected Regions

A connection may be thought of as an import. So region C imports from region B. The imports are filtered, so filter h may limit what region C sees from region B. Similarly region A imports from region B through filter f and region B imports from region A through filter g.

Unlike OSGi package imports between bundles, imports between regions are transitive. So region C can see bundles, exported packages, and services from region A, subject to filters g and h.

# 2 Basic Types

Some basic types need defining.

### Bundles

Bundles are identified by a bundle symbolic name and bundle version, but uniquely identified in the OSGi framework by a bundle location.

$$[BSN, BVer, BLoc]$$

### Packages

We abstract the notion of package version and the attributes associated with package imports and exports.

$$[Package]$$

### Services

We abstract all the details of services.

$$[Service]$$

### Regions

Regions are identified by a region identifer.

$$[RId]$$

## 3 Bundle

A bundle has a bundle symbolic name and bundle version, exports zero or more packages, and publishes zero or more services.

$$
\begin{array}{|l}
\textit{Bundle} \\
\hline
\mathit{name} : \mathit{BSN} \\
\mathit{version} : \mathit{BVer} \\
\mathit{location} : \mathit{BLoc} \\
\mathit{exportedPackages} : \mathbb{P}\,\mathit{Package} \\
\mathit{publishedServices} : \mathbb{P}\,\mathit{Service} \\
\hline
\end{array}
$$

We define some helper functions.

$$
\begin{array}{|l}
\mathit{bid} : \mathit{Bundle} \rightarrow (\mathit{BSN} \times \mathit{BVer}) \\
\hline
\mathit{bid} = (\lambda\, b : \mathit{Bundle} \bullet (b.\mathit{name}, b.\mathit{version}))
\end{array}
$$

$$
\begin{array}{|l}
\mathit{bundlePkgs} : \mathit{Bundle} \rightarrow \mathbb{P}\,\mathit{Package} \\
\hline
\mathit{bundlePkgs} = (\lambda\, b : \mathit{Bundle} \bullet b.\mathit{exportedPackages})
\end{array}
$$

$$
\begin{array}{|l}
\mathit{bundleSvcs} : \mathit{Bundle} \rightarrow \mathbb{P}\,\mathit{Service} \\
\hline
\mathit{bundleSvcs} = (\lambda\, b : \mathit{Bundle} \bullet b.\mathit{publishedServices})
\end{array}
$$

## 4 Region

A region contains a set of bundles each of which is uniquely identified *within the region* by its bundle symbolic name and bundle version and by its location.

A region also has a set of imported packages and a set of imported services which do not overlap the packages exported by and the services published by, respectively, the bundles in the region.

A region also contains some derived values: a function identifying the bundles in the region by symbolic name and version, a function identifying the bundles in the region by location, a set of the packages exported by bundles in the region (which is partitioned by bundle), a set of services published by bundles in the region (which is also partitioned by bundle), a combined set of local and imported packages, and a combined set of local and imported services.

$$
\begin{array}{|l}
\textit{Region} \\
\hline
\textit{bundles} : \mathbb{P}\,\textit{Bundle} \\
\textit{importedPkg} : \mathbb{P}\,\textit{Package} \\
\textit{importedSvc} : \mathbb{P}\,\textit{Service} \\
\\
\textit{nv} : \textit{BSN} \times \textit{BVer} \rightarrowtail\!\!\!\!\!\rightarrow \textit{Bundle} \\
\textit{l} : \textit{BLoc} \rightarrowtail\!\!\!\!\!\rightarrow \textit{Bundle} \\
\textit{localPkg} : \mathbb{P}\,\textit{Package} \\
\textit{localSvc} : \mathbb{P}\,\textit{Service} \\
\textit{pkg} : \mathbb{P}\,\textit{Package} \\
\textit{svc} : \mathbb{P}\,\textit{Service} \\
\hline
\textit{nv} = \{ b : \textit{bundles} \bullet (\textit{bid}\ b) \mapsto b \} \\
l = \{ b : \textit{bundles} \bullet b.\textit{location} \mapsto b \} \\
(\textit{bundlePkgs} \circ l)\ \mathsf{partition}\ \textit{localPkg} \land \\
(\textit{bundleSvcs} \circ l)\ \mathsf{partition}\ \textit{localSvc} \\
\langle \textit{localPkg}, \textit{importedPkg} \rangle\ \mathsf{partition}\ \textit{pkg} \\
\langle \textit{localSvc}, \textit{importedSvc} \rangle\ \mathsf{partition}\ \textit{svc} \\
\hline
\end{array}
$$

A bundle can be added to a region provided the region does not already contain a bundle with the given bundle's symbolic name and bundle version or a bundle with the given bundle's location. The bundles exported packages and published services must not overlap the region's packages and the region's services, respectively.

$$
\begin{array}{|l}
\textit{AddBundleOk} \\
\hline
\Delta \textit{Region} \\
b? : \textit{Bundle} \\
\hline
(\textit{bid}\ b?) \notin \operatorname{dom} \textit{nv} \\
b?.\textit{location} \notin \operatorname{dom} l \\
\textit{bundles}' = \textit{bundles} \cup \{ b? \} \\
\textit{importedPkg}' = \textit{importedPkg} \\
\textit{importedSvc}' = \textit{importedSvc} \\
\hline
\end{array}
$$

We define some helper functions.

$$\begin{array}{|l} \mathit{regionBundles} : \mathit{Region} \rightarrow \mathbb{P}\,\mathit{Bundle} \\ \hline \mathit{regionBundles} = (\lambda\, r : \mathit{Region} \bullet r.\mathit{bundles}) \end{array}$$

$$\begin{array}{|l} \mathit{regionPkgs} : \mathit{Region} \rightarrow \mathbb{P}\,\mathit{Package} \\ \hline \mathit{regionPkgs} = (\lambda\, r : \mathit{Region} \bullet r.\mathit{pkg}) \end{array}$$

$$\begin{array}{|l} \mathit{regionSvcs} : \mathit{Region} \rightarrow \mathbb{P}\,\mathit{Service} \\ \hline \mathit{regionSvcs} = (\lambda\, r : \mathit{Region} \bullet r.\mathit{svc}) \end{array}$$

$$\begin{array}{|l} \mathit{regionLoc} : \mathit{Region} \rightarrow (\mathit{BLoc} \rightarrowtail\!\!\!\!\!\!\shortmid\;\; \mathit{Bundle}) \\ \hline \mathit{regionLoc} = (\lambda\, r : \mathit{Region} \bullet r.l) \end{array}$$

We define a nil region.

$$\begin{array}{|l} \mathit{NIL} : \mathit{Region} \\ \hline \mathit{NIL}.\mathit{bundles} = \emptyset \\ \mathit{NIL}.\mathit{importedPkg} = \emptyset \\ \mathit{NIL}.\mathit{importedSvc} = \emptyset \end{array}$$

Regions with consistent partitioning may be combined to produce a composite region using the infix $\sqcup$ operator.

$$\begin{array}{|l} \_ \sqcup \_ : \mathit{Region} \times \mathit{Region} \mathrel{+\!\!\!\rightarrow} \mathit{Region} \\ \hline (\_ \sqcup \_) = \{r1, r2, r3 : \mathit{Region} \mid \\ \quad r3.\mathit{nv} = r1.\mathit{nv} \cup r2.\mathit{nv} \land \\ \quad r3.l = r1.l \cup r2.l \land \\ \quad r3.\mathit{pkg} = r1.\mathit{pkg} \cup r2.\mathit{pkg} \land \\ \quad r3.\mathit{svc} = r1.\mathit{svc} \cup r2.\mathit{svc} \bullet \\ \qquad ((r1, r2), r3)\} \end{array}$$

$\sqcup$ is idempotent, commutative, and associative and *NIL* acts as a zero.

$$\begin{array}{l} \vdash \forall\, r, s, t : \mathit{Region} \mid \{(r, s), (s, t)\} \subseteq \mathrm{dom}(\_ \sqcup \_) \bullet \\ \quad (r, r) \in \mathrm{dom}(\_ \sqcup \_) \land r \sqcup r = r \land \\ \quad r \sqcup s = s \sqcup r \land \\ \quad r \sqcup (s \sqcup t) = (r \sqcup s) \sqcup t \land \\ \quad (r, \mathit{NIL}) \in \mathrm{dom}(\_ \sqcup \_) \land r \sqcup \mathit{NIL} = \mathit{NIL} \end{array}$$

Since $\sqcup$ is commutative and associative, so we define a distributed form.

$$\begin{array}{|l} \bigsqcup : \mathbb{F}\,\mathit{Region} \mathrel{+\!\!\!\rightarrow} \mathit{Region} \\ \hline \bigsqcup \emptyset = \mathit{NIL} \land \\ (\forall\, r : \mathit{Region};\ f : \mathbb{F}\,\mathit{Region} \bullet \bigsqcup(\{r\} \cup f) = r \sqcup \bigsqcup f) \end{array}$$

## 5 Multiple Regions

A system of multiple regions has an indexed collection of regions, a set of bundles known to the system, a set of exported packages, and a set of services. The regions partition the set of all known bundles, the exported packages, and the services. Bundles in the system of multiple regions are uniquely identified by their location.

The system also have a convenience function for determining a bundle's region identifier and a convenience set containing all the valid region identifiers.

*Regions*

$$
\begin{array}{l}
reg : RId \pfun Region \\
allBundles : \mathbb{P}\, Bundle \\
allPackages : \mathbb{P}\, Package \\
allServices : \mathbb{P}\, Service \\
ls : BLoc \pinj Bundle \\
breg : Bundle \pfun RId \\
rids : \mathbb{P}\, RId \\
\hline
(regionBundles \circ reg)\ \mathsf{partition}\ allBundles \\
(regionPkgs \circ reg)\ \mathsf{partition}\ allPackages \\
(regionSvcs \circ reg)\ \mathsf{partition}\ allServices \\
breg = \{ b : allBundles;\ rid : \mathrm{dom}\, reg \mid b \in regionBundles\,(reg\ rid) \bullet (b, rid) \} \\
rids = \mathrm{dom}\, reg \\
(regionLoc \circ reg)\ \mathsf{partition}\ ls
\end{array}
$$

Each bundle in the system has an associated region.

$$Regions \vdash \mathrm{dom}\, breg = allBundles$$

This follows easily from $(regionBundles \circ reg)\ \mathsf{partition}\ allBundles$.

We expose the convenience function as an operation for determining a bundle's region. This function is well defined thanks to the preceding theorem.

*GetRegionOk*

$$
\begin{array}{l}
\Xi Regions \\
b? : Bundle \\
r! : Region \\
\hline
b? \in allBundles \\
r! = reg(breg\ b?)
\end{array}
$$

## 6 Filters

A filter specifies sets of bundles, exported packages, and services.

$$
\begin{array}{l}
\textit{Filter} \\
\hline
bf : \mathbb{P}(BSN \times BVer) \\
pf : \mathbb{P}\, Package \\
sf : \mathbb{P}\, Service
\end{array}
$$

We define some helper functions to perform filtering.

$$
\begin{array}{l}
xb : Bundle \times Filter \rightarrow Bundle \\
\hline
xb = (\lambda\, b : Bundle;\ f : Filter \bullet \\
\quad (\mu\, Bundle \mid \\
\quad\quad name = b.name \land \\
\quad\quad version = b.version \land \\
\quad\quad location = b.location \land \\
\quad\quad exportedPackages = b.exportedPackages \cap f.pf \land \\
\quad\quad publishedServices = b.publishedServices \cap f.sf))
\end{array}
$$

$$
\begin{array}{l}
fb : \mathbb{P}\, Bundle \times Filter \rightarrow \mathbb{P}\, Bundle \\
\hline
fb = (\lambda\, bs : \mathbb{P}\, Bundle;\ f : Filter \bullet \{ b : bs \mid (bid\ b) \in f.bf \bullet xb(b, f) \})
\end{array}
$$

$$
\begin{array}{l}
fp : \mathbb{P}\, Package \times Filter \rightarrow \mathbb{P}\, Package \\
\hline
fp = \{ ps : \mathbb{P}\, Package;\ f : Filter \bullet ((ps, f), ps \cap f.pf) \}
\end{array}
$$

$$
\begin{array}{l}
fs : \mathbb{P}\, Service \times Filter \rightarrow \mathbb{P}\, Service \\
\hline
fs = \{ ss : \mathbb{P}\, Service;\ f : Filter \bullet ((ss, f), ss \cap f.sf) \}
\end{array}
$$

We also define the most permissive filter.

$$
\begin{array}{l}
TOP : Filter \\
\hline
TOP.bf = BSN \times BVer \\
TOP.pf = Package \\
TOP.sf = Service
\end{array}
$$

We define an infix $\wr$ operator to apply a filter to a region and produce another region.

$$
\begin{array}{l}
\_ \wr \_ : Region \times Filter \nrightarrow Region \\
\hline
(\_ \wr \_) = \{ r1, r2 : Region;\ f : Filter \mid \\
\quad r2.bundles = \{ b : r1.bundles \mid (bid\ b) \in f.bf \bullet xb(b, f) \} \land \\
\quad r2.nv = f.bf \lhd r1.nv \land \\
\quad r2.l = r2.nv \circ bid \circ r1.l \land \\
\quad r2.pkg = r1.pkg \cap f.pf \land \\
\quad r2.svc = r1.svc \cap f.sf \bullet \\
\quad\quad ((r1, f), r2) \}
\end{array}
$$

The most permissive filter can be applied to any region without effect.

$$\vdash \forall\, r : \mathit{Region} \bullet r \wr \mathit{TOP} = r$$

## 7 Connected Regions

Regions are connected by filters. Every region is connected to itself by the most permissive filter. So the bundles in a region can see all the bundles, exported packages, and services in that region.

*ConnectedRegions*

$$
\begin{array}{l}
\mathit{Regions} \\
\mathit{filter} : \mathit{RId} \times \mathit{RId} \pfun \mathit{Filter} \\
\mathit{squash} : \mathit{RId} \pfun \mathit{Region} \\
\hline
\mathit{first} \limg \operatorname{dom} \mathit{filter} \rimg = \mathit{rids} \\
\mathit{second} \limg \operatorname{dom} \mathit{filter} \rimg = \mathit{rids} \\
(\forall \mathit{rid} : \mathit{rids} \bullet \mathit{filter}(\mathit{rid}, \mathit{rid}) = \mathit{TOP}) \\
\operatorname{dom} \mathit{squash} = \mathit{rids} \\
(\forall \mathit{rid} : \mathit{rids} \bullet \\
\qquad (\mathbf{let}\ r == \mathit{reg}\ \mathit{rid} \bullet \\
\qquad\qquad \mathit{squash}(\mathit{rid}) = r \sqcup \bigsqcup\{r2 : \mathit{RId} \mid (\mathit{rid}, r2) \in \operatorname{dom} \mathit{filter} \land r2 \neq \mathit{rid} \bullet \\
\qquad\qquad\qquad \mathit{squash}(r2) \wr \mathit{filter}(\mathit{rid}, r2)\}))
\end{array}
$$

## 8 Hooks

We now describe the behaviour of the hooks in terms of the system of connected regions.

$$
\begin{array}{|l}
\textit{BundleFindHook} \\
\hline
\Xi \mathit{ConnectedRegions} \\
\mathit{finder}? : \mathit{Bundle} \\
\mathit{candidates}? : \mathbb{P}\, \mathit{Bundle} \\
\mathit{found}! : \mathbb{P}\, \mathit{Bundle} \\
\hline
\mathit{finder}? \in \mathit{allBundles} \\
\mathit{found}! = \mathit{candidates}? \cap (\mathit{squash}(\mathit{breg}\ \mathit{finder}?)).\mathit{bundles} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\textit{BundleEventHook} \\
\hline
\Xi \mathit{ConnectedRegions} \\
\mathit{listeners}? : \mathbb{P}\, \mathit{Bundle} \\
\mathit{eb}? : \mathit{Bundle} \\
\mathit{fl}! : \mathbb{P}\, \mathit{Bundle} \\
\hline
\mathit{listeners}? \subseteq \mathit{allBundles} \\
\mathit{fl}! = \{ l : \mathit{listeners}? \mid \mathit{eb}? \in (\mathit{squash}(\mathit{breg}\ l)).\mathit{bundles} \} \\
\hline
\end{array}
$$

## 9 Z Notation

Numbers:

$\mathbb{N}$ `Natural numbers {0,1,...}`

Propositional logic and the schema calculus:

| | | | |
|---|---|---|---|
| $\ldots \land \ldots$ | `And` | $\langle\!\langle \ldots \rangle\!\rangle$ | `Free type injection` |
| $\ldots \lor \ldots$ | `Or` | $[\ldots]$ | `Given sets` |
| $\ldots \Rightarrow \ldots$ | `Implies` | $', ?, !,_0 \ldots_9$ | `Schema decorations` |
| $\forall .. \mid .. \bullet ..$ | `For all` | $\ldots \vdash \ldots$ | `theorem` |
| $\exists .. \mid .. \bullet ..$ | `There exists` | $\theta \ldots$ | `Binding formation` |
| $\ldots \setminus \ldots$ | `Hiding` | $\lambda \ldots$ | `Function definition` |
| $\ldots \widehat{=} \ldots$ | `Schema definition` | $\mu \ldots$ | `Mu-expression` |
| $\ldots == \ldots$ | `Abbreviation` | $\Delta \ldots$ | `State change` |
| $\ldots ::= \ldots \mid \ldots$ | `Free type definition` | $\Xi \ldots$ | `Invariant state change` |

Sets and sequences:

| | | | |
|---|---|---|---|
| $\{\ldots\}$ | `Set` | $\ldots \setminus \ldots$ | `Set difference` |
| $\{.. \mid .. \bullet ..\}$ | `Set comprehension` | $\bigcup \ldots$ | `Distributed union` |
| $\mathbb{P} \ldots$ | `Set of subsets of` | $\# \ldots$ | `Cardinality` |
| $\emptyset$ | `Empty set` | $\ldots \subseteq \ldots$ | `Subset` |
| $\ldots \times \ldots$ | `Cartesian product` | $\ldots \subset \ldots$ | `Proper subset` |
| $\ldots \in \ldots$ | `Set membership` | $\ldots$ partition $\ldots$ | `Set partition` |
| $\ldots \notin \ldots$ | `Set non-membership` | seq | `Sequences` |
| $\ldots \cup \ldots$ | `Union` | $\langle \ldots \rangle$ | `Sequence` |
| $\ldots \cap \ldots$ | `Intersection` | disjoint $\ldots$ | `Disjoint sequence of sets` |

Functions and relations:

| | | | |
|---|---|---|---|
| $\ldots \leftrightarrow \ldots$ | `Relation` | $\ldots \mapsto \ldots$ | `maplet` |
| $\ldots ⇸ \ldots$ | `Partial function` | $\ldots^{\sim}$ | `Relational inverse` |
| $\ldots \rightarrow \ldots$ | `Total function` | $\ldots^{*}$ | `Reflexive-transitive closure` |
| $\ldots ⤔ \ldots$ | `Partial injection` | $\ldots \llparenthesis \ldots \rrparenthesis$ | `Relational image` |
| $\ldots \rightarrowtail \ldots$ | `Injection` | $\ldots \oplus \ldots$ | `Functional overriding` |
| dom $\ldots$ | `Domain` | $\ldots \lhd \ldots$ | `Domain restriction` |
| ran $\ldots$ | `Range` | $\ldots ⩤ \ldots$ | `Domain subtraction` |

Axiomatic descriptions:

*Declarations*

*Predicates*

Schema definitions:

*SchemaName*

*Declaration*

*Predicates*