Draft

Regions in Virgo (v0.3)

Glyn Normington
Steve Powell

January 12, 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".

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 *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.

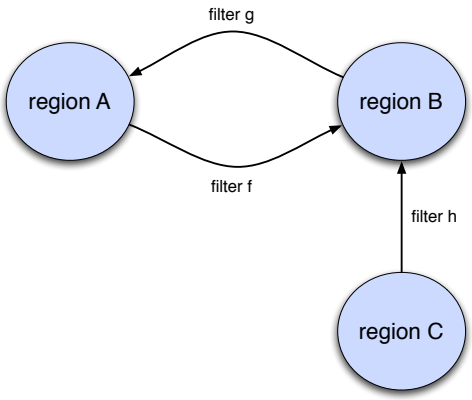

Figure 1: Connected Regions

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.

1.1 Resolver Hook

The resolver hook limits the exported packages that the bundles in a given region may wire to, depending on the region containing the bundle that exports each candidate exported package and the filters between the regions.

For example, Figure 1.1 shows a bundle Z being resolved which imports packages p and q. Bundle X in region B exports both p and q while bundle Y in region B exports only p.

Region A is connected to region B with a filter that allows only package p to be seen by region A. The net effect is that the import of p may be satisfied by either bundle X or Y but the import of package q may not be satisfied by bundle X since q is filtered out.

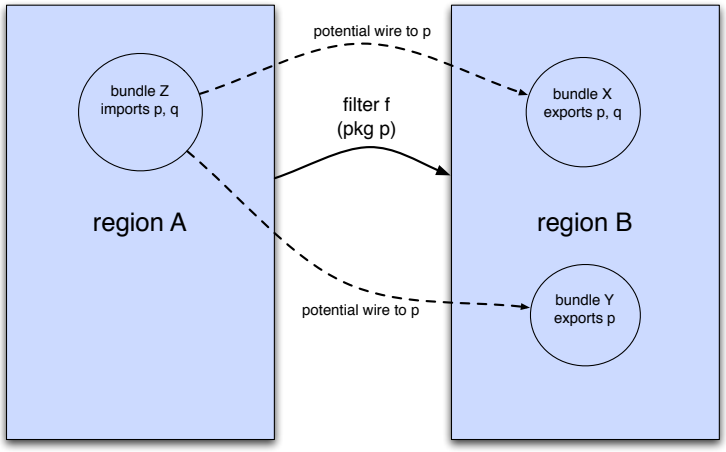

Figure 2: Package Filtering

Another example in Figure 1.1 shows a package p transitively visible through two filters via an intermediate region.

Bundle Z may wire to bundle X or bundle Y for package p, but not for packages q and r which are both filtered out on the way from C to A.

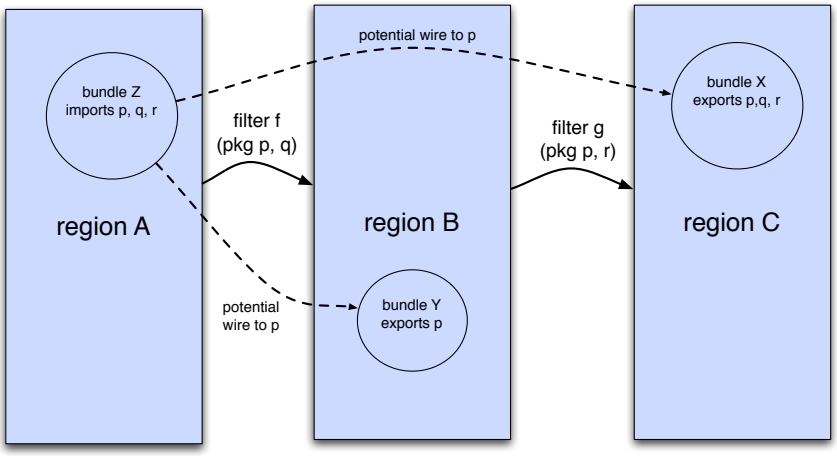

Figure 3: Transitive Package Filtering

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.

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

It is unusual, and usually bad practice, to include the primary key (in this case the location) of an entity in the entity's schema. We do it here to ensure that all bundles are distinct and use this property later when determining which region a bundle belongs to.

We define a helper function for extracting a pair consisting of the bundle symbolic name and bundle version from a bundle.

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

4 Region

We build up to a definition of a region in several small steps.

4.1 Proto-Region

A proto-region contains a set of bundles.

$$\begin{array}{l} \textbf{schema } \mathit{ProtoRegion} \\ \quad \mathit{bundles} : \mathbb{P}\, \mathit{Bundle} \end{array}$$

4.2 Indexed Region

An indexed region adds some indices, and thereby some constraints, to a proto-region.

An indexed 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.

The indices are a function identifying the bundles in the region by symbolic name and version and a function identifying the bundles in the region by location.

$$\begin{array}{l} \textbf{schema } \mathit{IndexedRegion} \\ \quad \mathit{ProtoRegion} \\ \quad \mathit{nv} : \mathit{BSN} \times \mathit{BVer} \rightarrowtail\!\!\!\!\!\shortmid\; \mathit{Bundle} \\ \quad l : \mathit{BLoc} \rightarrowtail\!\!\!\!\!\shortmid\; \mathit{Bundle} \\ \hline \quad \mathit{nv} = \{ b : \mathit{bundles} \bullet (\mathit{bid}\ b) \mapsto b \} \\ \quad l = \{ b : \mathit{bundles} \bullet b.\mathit{location} \mapsto b \} \end{array}$$

4.3 OpenRegion

An open region is an indexed region with some derived sets of packages exported by bundles in the region and services published by bundles in the region.

$$\begin{array}{l} \textbf{schema } \mathit{OpenRegion} \\ \quad \mathit{IndexedRegion} \\ \quad \mathit{localPkg} : \mathbb{P}\, \mathit{Package} \\ \quad \mathit{localSvc} : \mathbb{P}\, \mathit{Service} \\ \hline \quad \mathit{localPkg} = \bigcup \{ b : \mathit{bundles} \bullet b.\mathit{exportedPackages} \} \\ \quad \mathit{localSvc} = \bigcup \{ b : \mathit{bundles} \bullet b.\mathit{publishedServices} \} \end{array}$$

4.4 LinkedRegion

A linked region is an indexed region with additional sets of imported packages and services.

$$
\begin{array}{|l}
\textit{LinkedRegion} \\
\hline
\textit{OpenRegion} \\
\textit{importedPkg} : \mathbb{P}\, \textit{Package} \\
\textit{importedSvc} : \mathbb{P}\, \textit{Service} \\
\hline
\end{array}
$$

4.5 Region

A region is a linked region with derived sets of all packages and all services.

$$
\begin{array}{|l}
\textit{Region} \\
\hline
\textit{LinkedRegion} \\
\textit{pkg} : \mathbb{P}\, \textit{Package} \\
\textit{svc} : \mathbb{P}\, \textit{Service} \\
\hline
\textit{pkg} = \textit{localPkg} \cup \textit{importedPkg} \\
\textit{svc} = \textit{localSvc} \cup \textit{importedSvc} \\
\hline
\end{array}
$$

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.

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

We define a nil region.

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

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

$$
\begin{array}{|l}
\_ \sqcup \_ : \textit{Region} \times \textit{Region} \pfun \textit{Region} \\
\hline
(\_ \sqcup \_) = (\lambda\, r1, r2 : \textit{Region} \bullet \\
\qquad (\mu\, \textit{Region} \mid \\
\qquad\quad \textit{bundles} = r1.\textit{bundles} \cup r2.\textit{bundles} \land \\
\qquad\quad \textit{pkg} = r1.\textit{pkg} \cup r2.\textit{pkg} \land \\
\qquad\quad \textit{svc} = r1.\textit{svc} \cup r2.\textit{svc}))
\end{array}
$$

The precondition of $\sqcup$ is that the input regions have consistent bundle spaces, that is no bundle in one region has the same symbolic name and version as a bundle in the other region. Also, no bundle in one region has the same location as a bundle in the other region.

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

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

We define the set of all pairwise consistent regions

$$\mathit{ConsistentRegionPairs} == \mathrm{dom}(\_ \sqcup \_)$$

and the set of all pairwise consistent finite sets of regions.

$$\mathit{ConsistentRegionSets} == \{f : \mathbb{F}\, \mathit{Region} \mid \forall\, r, s : f \bullet (r,s) \in \mathit{ConsistentRegionPairs}\}$$

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

$$\begin{array}{|l}
\bigsqcup : \mathbb{F}\, \mathit{Region} \nrightarrow \mathit{Region} \\
\hline
\bigsqcup \varnothing = \mathit{NIL} \land \\
(\forall\, r : \mathit{Region};\ f : \mathit{ConsistentRegionSets} \mid \{r\} \cup f \in \mathit{ConsistentRegionSets} \bullet \\
\qquad \bigsqcup(\{r\} \cup f) = r \sqcup \bigsqcup f)
\end{array}$$

5 Multiple Regions

A system of multiple regions has an indexed collection of regions, a convenience of all the region identifiers in the system, and a function for determining the region identifier of any bundle in the system.

Regions

$$
\begin{array}{l}
reg : RId \pfun Region \\
allBundles : \mathbb{P}\, Bundle \\
rids : \mathbb{P}\, RId \\
breg : Bundle \pfun RId \\
\hline
allBundles = \bigcup \{ r : \operatorname{ran} reg \bullet r.bundles \} \\
rids = \operatorname{dom} reg \\
breg = \{ b : Bundle;\ rid : rids \mid b \in (reg\ rid).bundles \bullet (b, rid) \}
\end{array}
$$

Since *breg* is a function, no bundle can belong to more than one region.

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.

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

We define some helper functions to perform filtering.

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

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

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

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

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.

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

$\wr$ is total since packages and services not filtered out which are exported or published by a bundle which *is* filtered out end up in the resultant region's imported packages and imported services sets, respectively.

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

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

Filters can be applied in any order with the same effect.

$$\vdash \forall\, r : \mathit{Region};\ f, g : \mathit{Filter} \bullet (r \wr f) \wr g = (r \wr g) \wr f$$

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

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

8 Hooks

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

BundleFindHook

$$\Xi ConnectedRegions$$
$$finder? : Bundle$$
$$candidates? : \mathbb{P}\, Bundle$$
$$found! : \mathbb{P}\, Bundle$$

$$finder? \in allBundles$$
$$found! = candidates? \cap (squash(breg\ finder?)).bundles$$

BundleEventHook

$$\Xi ConnectedRegions$$
$$listeners? : \mathbb{P}\, Bundle$$
$$eb? : Bundle$$
$$fl! : \mathbb{P}\, Bundle$$

$$listeners? \subseteq allBundles$$
$$fl! = \{l : listeners? \mid eb? \in (squash(breg\ l)).bundles\}$$

ResolveHookFilterMatches

$$\Xi ConnectedRegions$$
$$requirer? : Bundle$$
$$candidates? : \mathbb{P}\, Package$$
$$filtered! : \mathbb{P}\, Package$$

$$requirer? \in allBundles$$
$$filtered! = candidates? \cap (squash(breg\ requirer?)).pkg$$

ServiceFindHook

$$\Xi ConnectedRegions$$
$$finder? : Bundle$$
$$candidates? : \mathbb{P}\, Service$$
$$found! : \mathbb{P}\, Service$$

$$finder? \in allBundles$$
$$found! = candidates? \cap (squash(breg\ finder?)).svc$$

ServiceEventHook

$$\Xi ConnectedRegions$$
$$listeners? : \mathbb{P}\, Bundle$$
$$es? : Service$$
$$fl! : \mathbb{P}\, Bundle$$

$$listeners? \subseteq allBundles$$
$$fl! = \{l : listeners? \mid es? \in (squash(breg\ l)).svc\}$$

9 Z Notation

Numbers:

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

Propositional logic and the schema calculus:

$\ldots \wedge \ldots$	`And`	$\langle\!\langle \ldots \rangle\!\rangle$	`Free type injection`
$\ldots \vee \ldots$	`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`
$\ldots \mathrel{\widehat=} \ldots$	`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`
$\varnothing$	`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`
$\ldots \nrightarrow \ldots$	`Partial function`	$\ldots^{\sim}$	`Relational inverse`
$\ldots \rightarrow \ldots$	`Total function`	$\ldots^{*}$	`Reflexive-transitive closure`
$\ldots \rightarrowtail \ldots$	`Partial injection`	$\ldots(\!\mid \ldots \mid\!)$	`Relational image`
$\ldots \rightarrowtail \ldots$	`Injection`	$\ldots \oplus \ldots$	`Functional overriding`
$\mathrm{dom} \ldots$	`Domain`	$\ldots \lhd \ldots$	`Domain restriction`
$\mathrm{ran} \ldots$	`Range`	$\ldots \lhd \ldots$	`Domain subtraction`

Axiomatic descriptions:

Declarations

Predicates

Schema definitions:

SchemaName

Declaration

Predicates