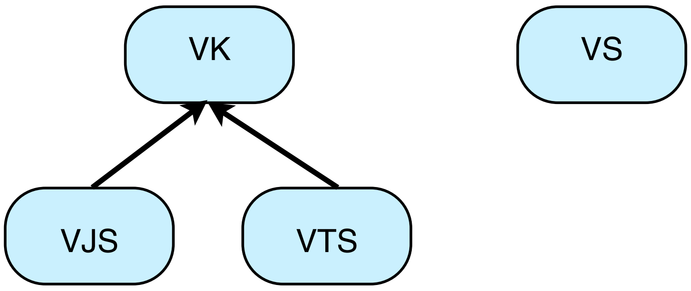
p2 repositories and Virgo release

Building the packaging repositories chain

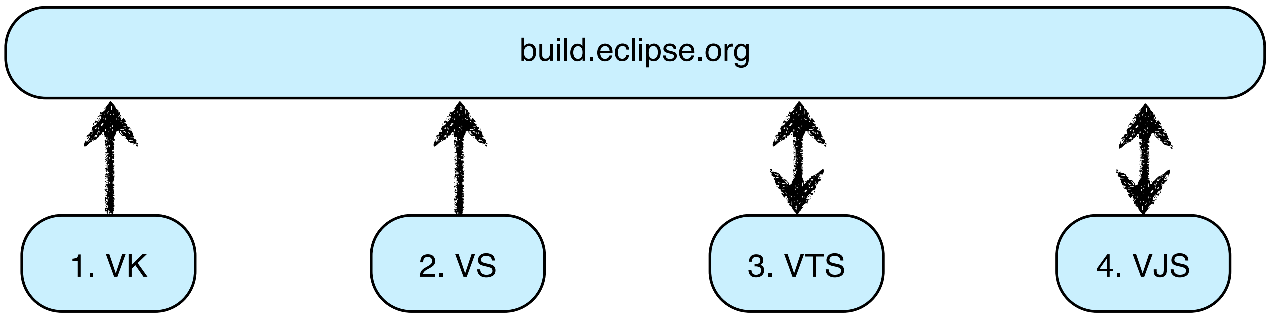
Today we have several packaging repositories - VK, VTS, VJS and VS.
These are built by a ruby script incrementally and ordered. VK serves as a base for VTS and VJS. VS is a separate packaging repository with no relation to other repositories. Important is the sharing and reusing aspect of the build process.



The build chain contains many component repositories that the packaging repositories assemble. Releasing this chain goes down an ordered path. At every packaging repository there is a stop that uploads the package results on every defined destination (Ivy, Maven, build.eclipse.org, download.eclipse.org).
A normal integration build publishes the packaging artifacts just to Ivy and build.eclipse.org.

Producing p2 repos from packaging repositories

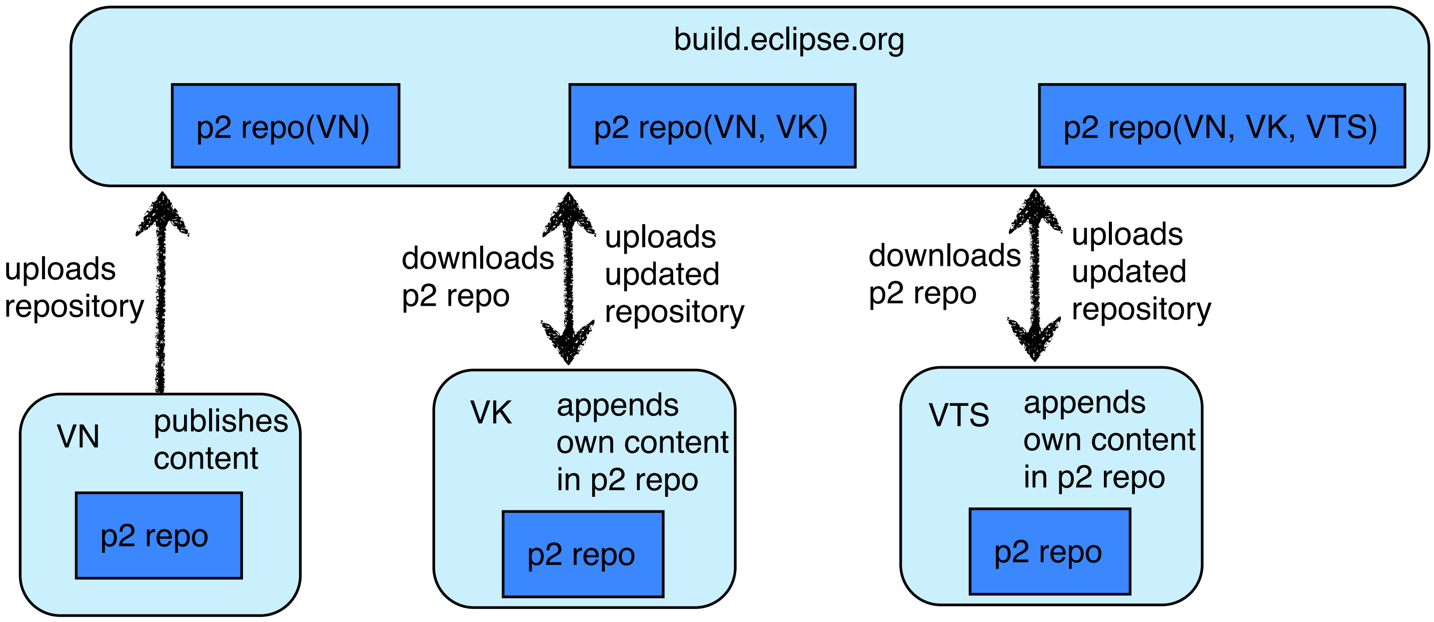
The interaction between the packaging repositories happens via build.eclipse.org. For example VTS and VJS fetch their VK base from build.eclipse.org then build on top of it and upload the results back. Non-packaging artifacts are published only to Ivy.
Here's how flow goes in the build chain



Let's take a look from the two different perspectives - integration build and release.

Integration build
Integration builds ensure the consistency of the product as a whole. A ripple through every packaging repository ends up with updated package results on build.eclipse.org. Which can be consumed by a repository further down the chain.

How to produce a p2 repository?
The proposed model is similar to the situation we have with the current zip publishing. The idea is to have a p2 repository updated by every packaging build. This way the content of each packaging repository will be appended to the existing and will reuse content from previous packaging repositories. Since the products of each next distribution in the chain inherits and uses the contents of the previous this model is convenient.
Here's how the proposed flow looks like



Each packaging repository uploads p2 repo in its own location. The hierarchy between the packaging repos is clear so the next repository in the chain is responsible to download the p2 repo of the previous repository.
The whole operation produces as many p2 repos as there are packaging repos. Each new p2 repository includes the previous and adds its own content to it.
Zips of the distributions can be produced by installing a product from the assembled p2 repository.

Release
Releases produce the same artifacts but there are also packages available in Maven and on download.eclipse.org.
The process of building p2 repos during release is more or less the same as the integration build. There are some differences though.
- Uploading to download.eclipse.org makes sense only for the last built p2 repository. It contains all products in the build chain allowing users to take any Virgo product they want from a single repository(url). This means we will have a ***separate p2 repository per release***.
- Uploading to Maven doesn't makes sense for the p2 repositories.
- Uploading of the zipped distributions should remain unchanged. They're all uploaded to download.eclipse.org.

Dismissed options

- Composing p2 repositories - Composing p2 repositories is the ability to have an umbrella repository that contains many child repositories which users see as a single one. Having this in our case just adds complexity over the single repository approach described above.
- Having every released version in one repository - That's a compelling case. It means a single "release" repository url - convenient for users. The reality is that this is a bit risky. In order to have such structure the product and its contents need to have strict requirements defined (no generic requirements - 0.0.0), otherwise we risk having modifiable already released products. Their generic requirements might get updated when the required component gets updated with a newer release.