Dependency Management for Datacenter Applications and their Resources

Dr. Oskar v. Dungern
2013-10-18

Introduction

It is shown how dependencies between revisions of datacenter applications and revisions of required resources are managed.

- First, a model is created consisting of
  - objects representing the applications,
  - others representing the resources and
  - the relations between specified revisions of applications and resources.
- The application objects are purely conceptual and represent a certain set of resource revisions, whereas the resource objects represent existing digital files.
- To alleviate searching and management, an object usually carries a number of attributes (description, category, status, ….), but this is not considered, here.
- On either side, a relation may be
  - ‚floating‘: this end points always to the newest revision of the respective object.
  - ‚fixed‘: this end continues to point to a specified revision of an object, even if the object is updated, i.e. a new revision is created.

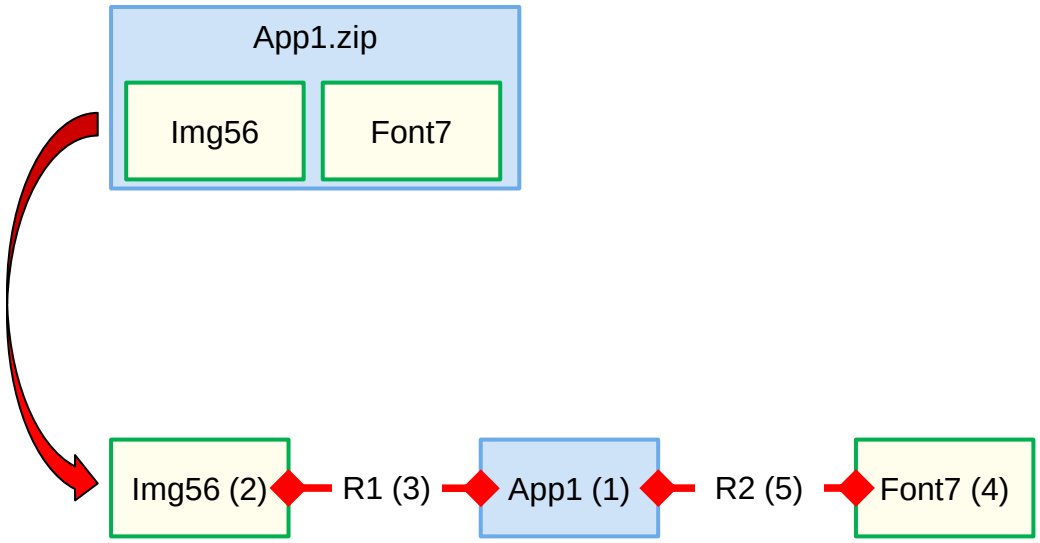
1. An application with 2 resources is updated
1.1 Overview

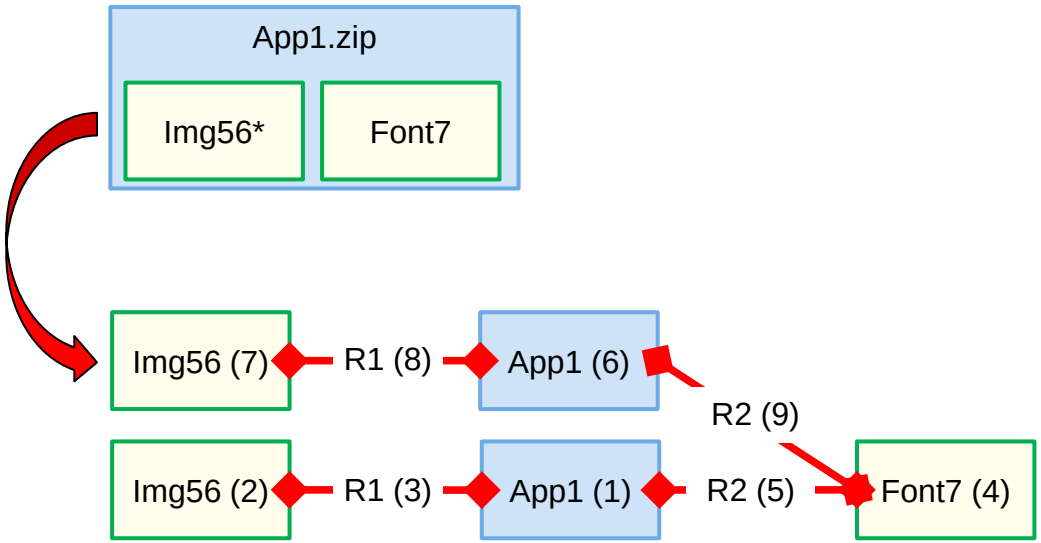
First step:

- A deployment package with two resources is handed over to the Resource Director
- Both resources are stored
- The Resource Director object model shows the dependencies.

Second step:

- A deployment package with the same application name consisting of an updated and an identical resource is handed over.
- The updated resource Img56* is stored
- The Resource Director object model shows the dependencies.



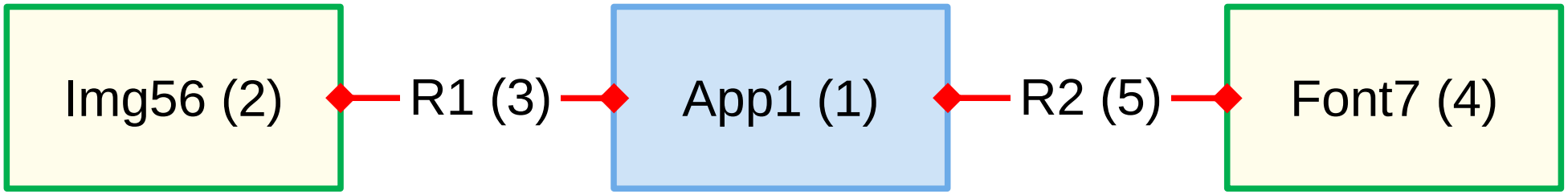


ReqIF Server

1. An application with 2 resources is updated
1.2 Initial Situation

Assume that an application needs two resources, an image and a font

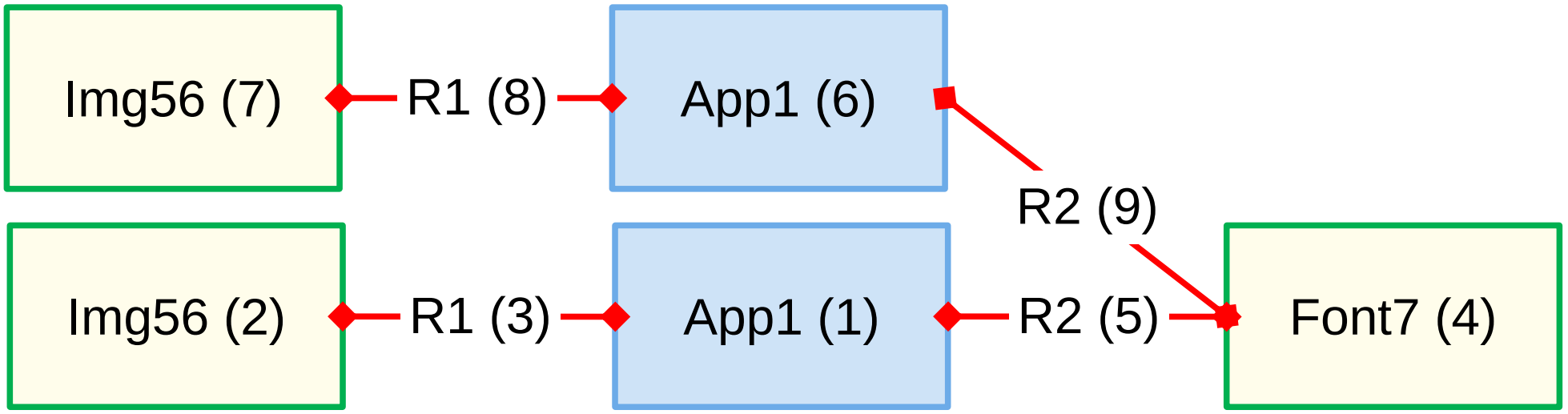
- The number in brackets is the transaction count = revision number.
- Relation R1 is specific to the revisions of both App1 and Img56 (‚fixed‘).
- Relation R2 is specific to the revisions of both App1 and Font7 (‚fixed‘).



1. An application with 2 resources is updated
1.3 Update the application and one of the resources

Suppose, the image is newer and the font is identical:

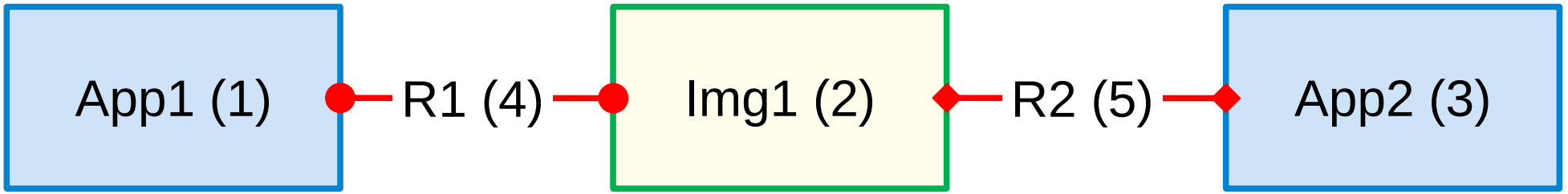
- A new revision of App1 and a new revision of Img56 are created
- Both relations are updated explicitly to reflect the dependencies
- Both application revisions can be retrieved with their respective resources.



2. Two applications share a resource
2.1 Initial Situation

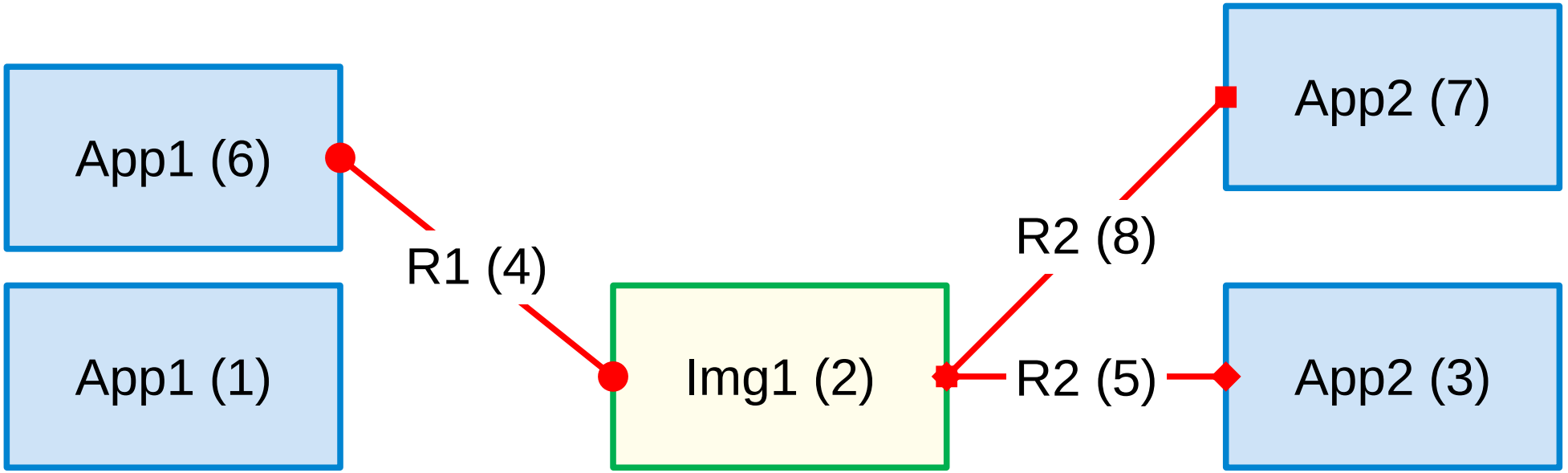
Assume that 2 applications need the same image resource

- The number in brackets is the transaction count = revision number.
- Relation R1 is unspecific to the revisions of both App1 and Img1 (‚floating‘).
- Relation R2 is specific to the revisions of App2 and Img1 (‚fixed‘).



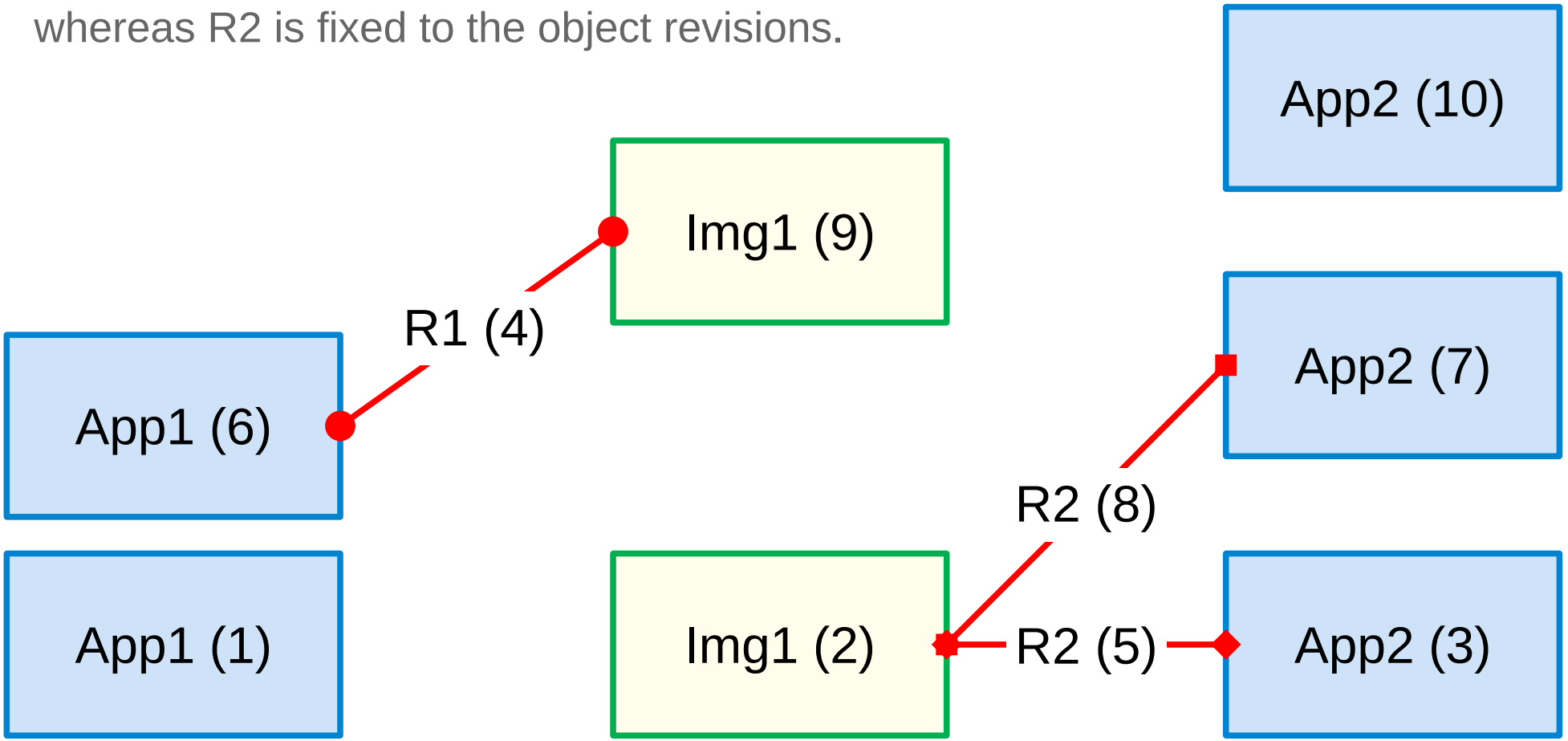
2. Two applications share a resource
2.2 Updating App1 and App2

- New revisions of both App1 and App2 are created.
- Being ‚floating‘, relation R1 automatically applies to the new revisions of App1.
- Being ‚fixed‘, R2 must be explicitly updated to relate App2 (7) and Img1 (2).



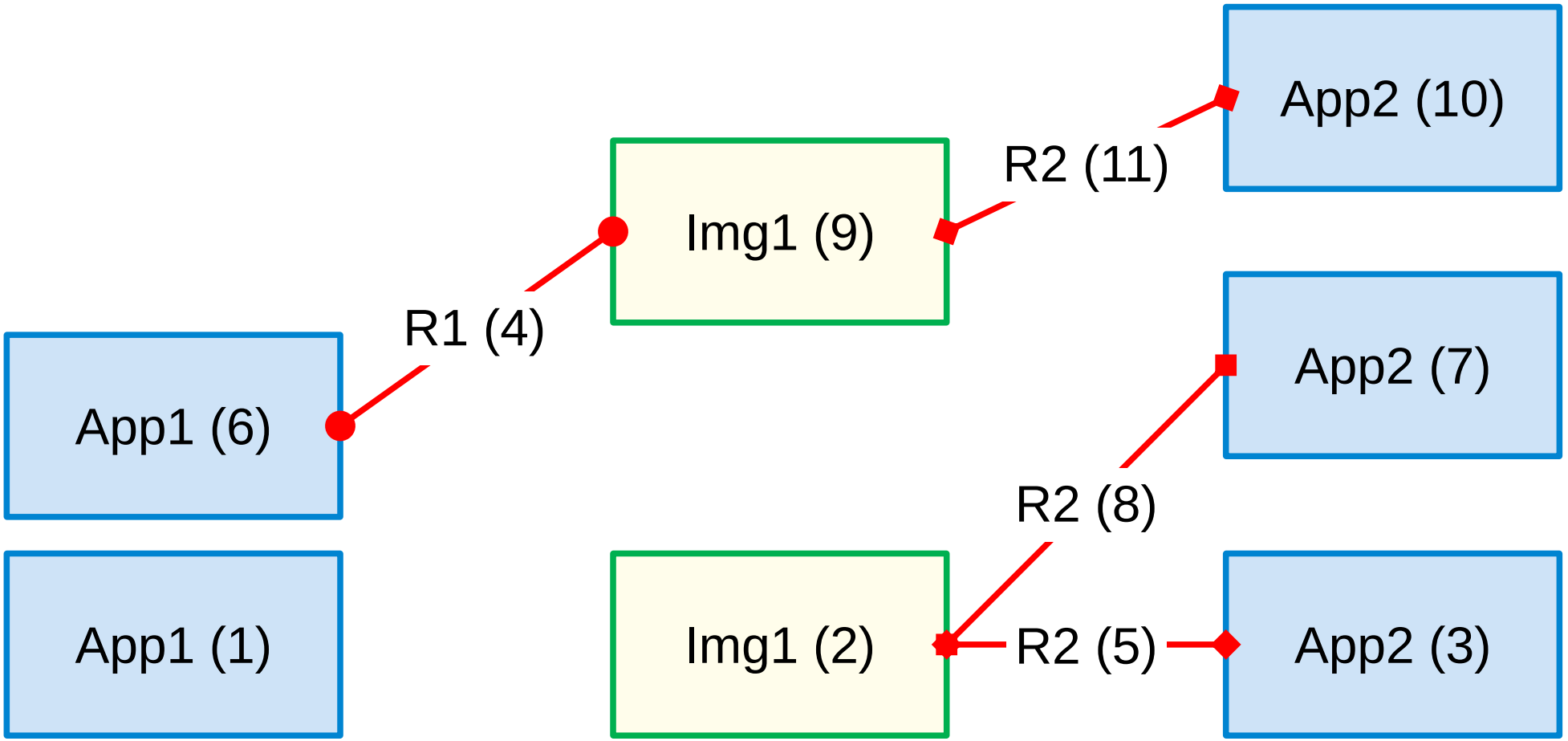
2. Two applications share a resource
2.3 Updating Img1 and App2

- Also here, being floating, relation R1 automatically applies to the newest revisions of the updated objects,
- whereas R2 is fixed to the object revisions.



2. Two applications share a resource
2.4 Updating R2

- Being ‚fixed‘, R2 must be explicitly updated to point to the newest revisions.



ReqIF Server

Is this interesting for you?

Any questions?

Contact
Dr.-Ing. Oskar v. Dungern
+49 173 670 9958
od@enso-managers.com

Information
www.reqf.de