Analysis of a Software Development Kit (SDK) for SPDX 2.0

By Jack Manbeck (Texas Instruments)

Version 0.2 21 March 2014

Document License

- This work is licensed under the Creative Commons Attribution License 3.0 Unported (CC-BY-3.0)

Contributors to this document

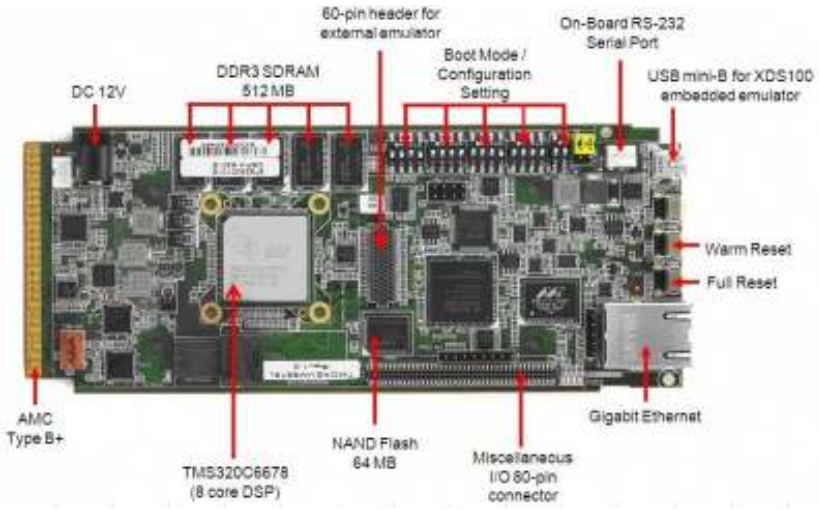
- Copyright (C) 2014 Texas Instruments Incorporated - http://www.ti.com/

Overview

- This is an analysis of an SDK that TI delivers to help flesh out the SPDX 2.0 presentation with respect to relationships.
- The entire SDK will not be documented as its too large but enough should be shown to get the “flavor”.
- This example should be applicable to use case 10 (2, 3,4, 6 and 7); http://wiki.spdx.org/view/Technical_Team/Use_Cases/2.0
  - Other versions could include use case 9 as well

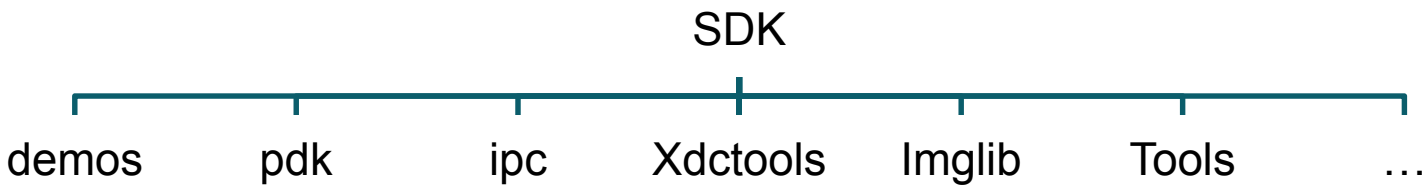
What is an SDK?

- Start with a Reference Design that has one of our Chips on a board along with peripheral interfaces, etc.,.
- We then do a port of an operating system, e.g. Linux
- We add documentation, tools for development, drivers, frameworks and other components to create a Software Development Kit (SDK)
- Then we add application to demonstrate the capabilities of the platform using this SDK
- SDK releases can be made up of tens of thousand of files and hundreds of applications



Reference Design or Evaluation Module (EVM)

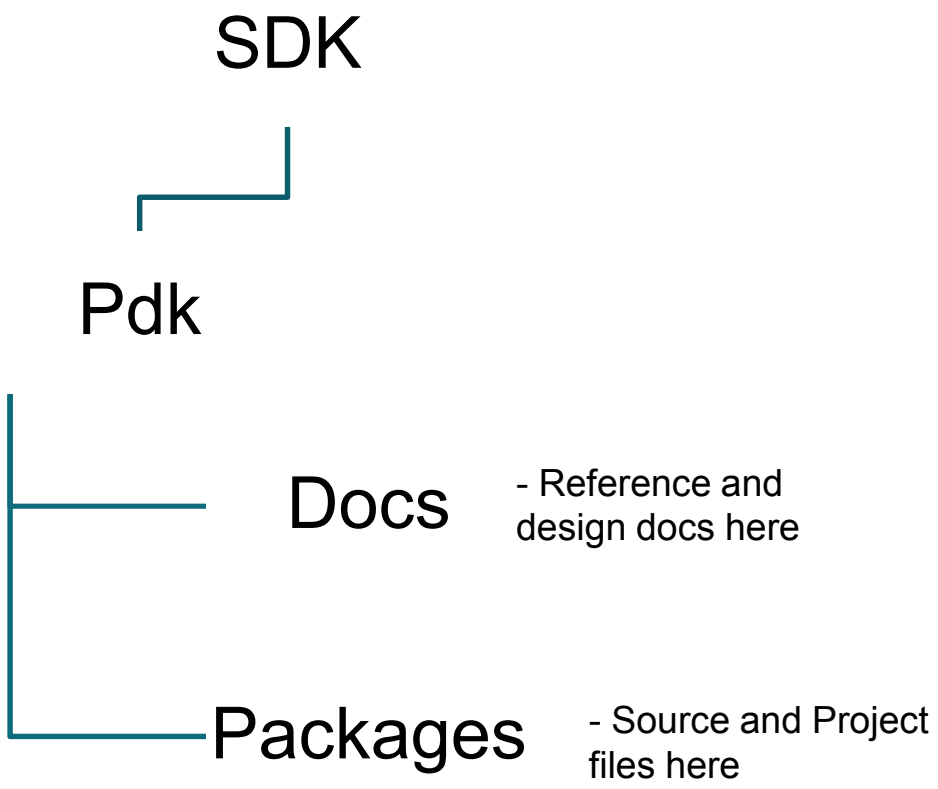
Example SDK for this Analysis



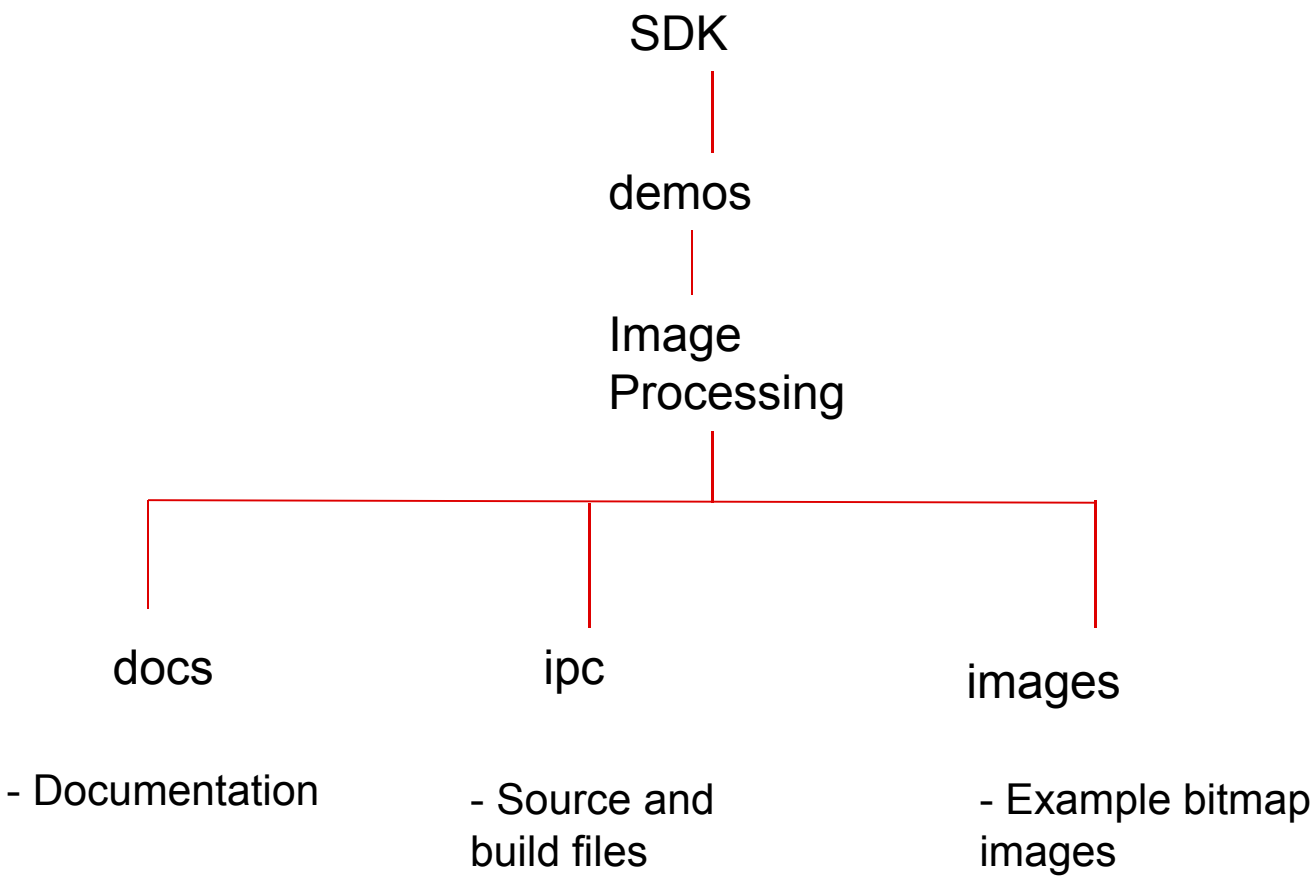
The entire SDK will not be documented as its too large but enough should be shown to get the “flavor”.

- About a 1 Gig download
- Has libraries and demos/examples
- Has tools that run on a host PC for development or on an EVM
- There are additional downloads

Partial breakdown of SDK PDK Directory



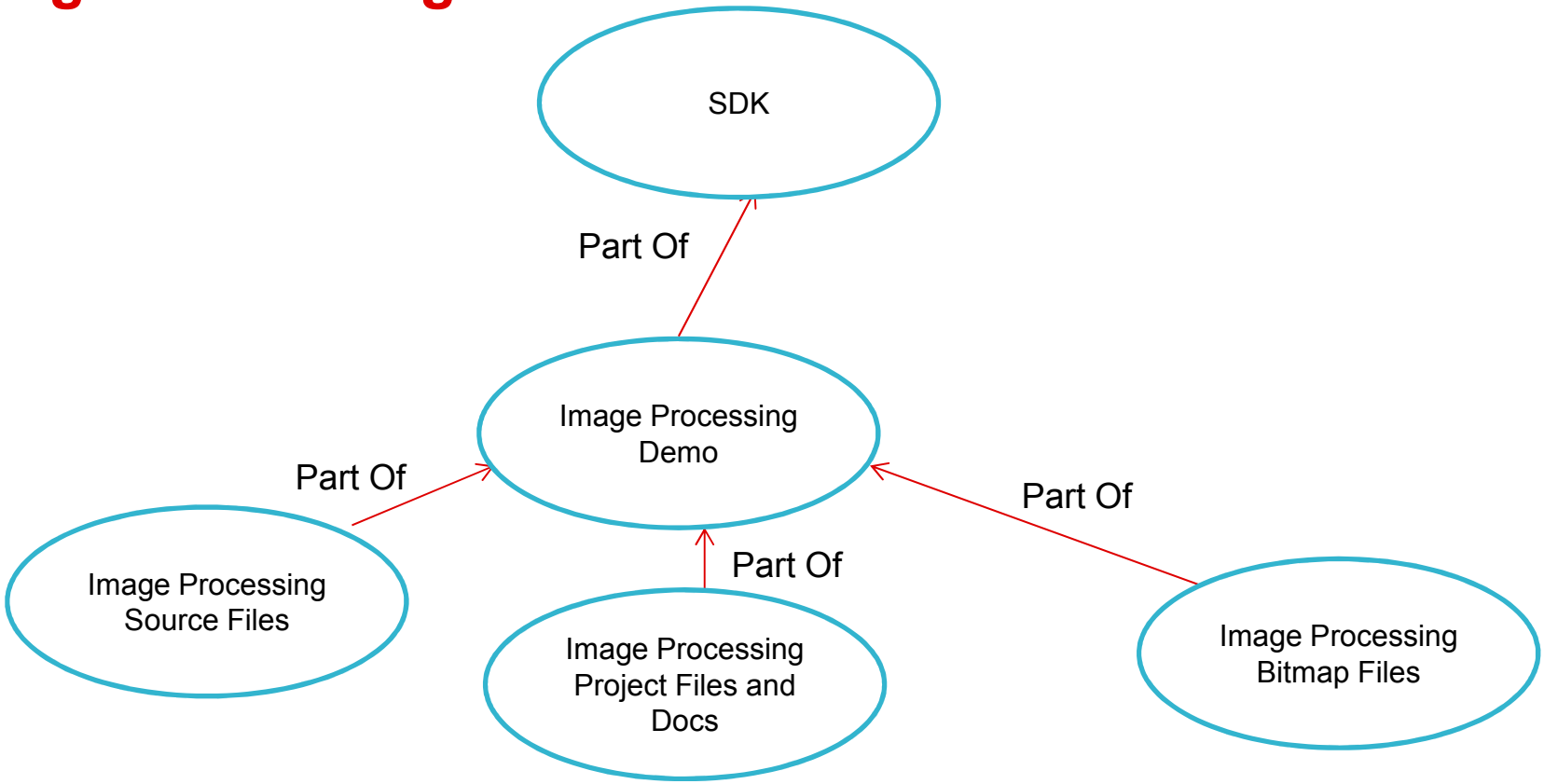
Partial breakdown of MCSDK Directory



Lets look at the Image Processing Demo

- There is documentation, some .bmp files for the application to process, and then different builds for different models.
- The binary when built from this source includes several libraries (compiler used, Ipc library, device drivers from the PDK, etc.,.)

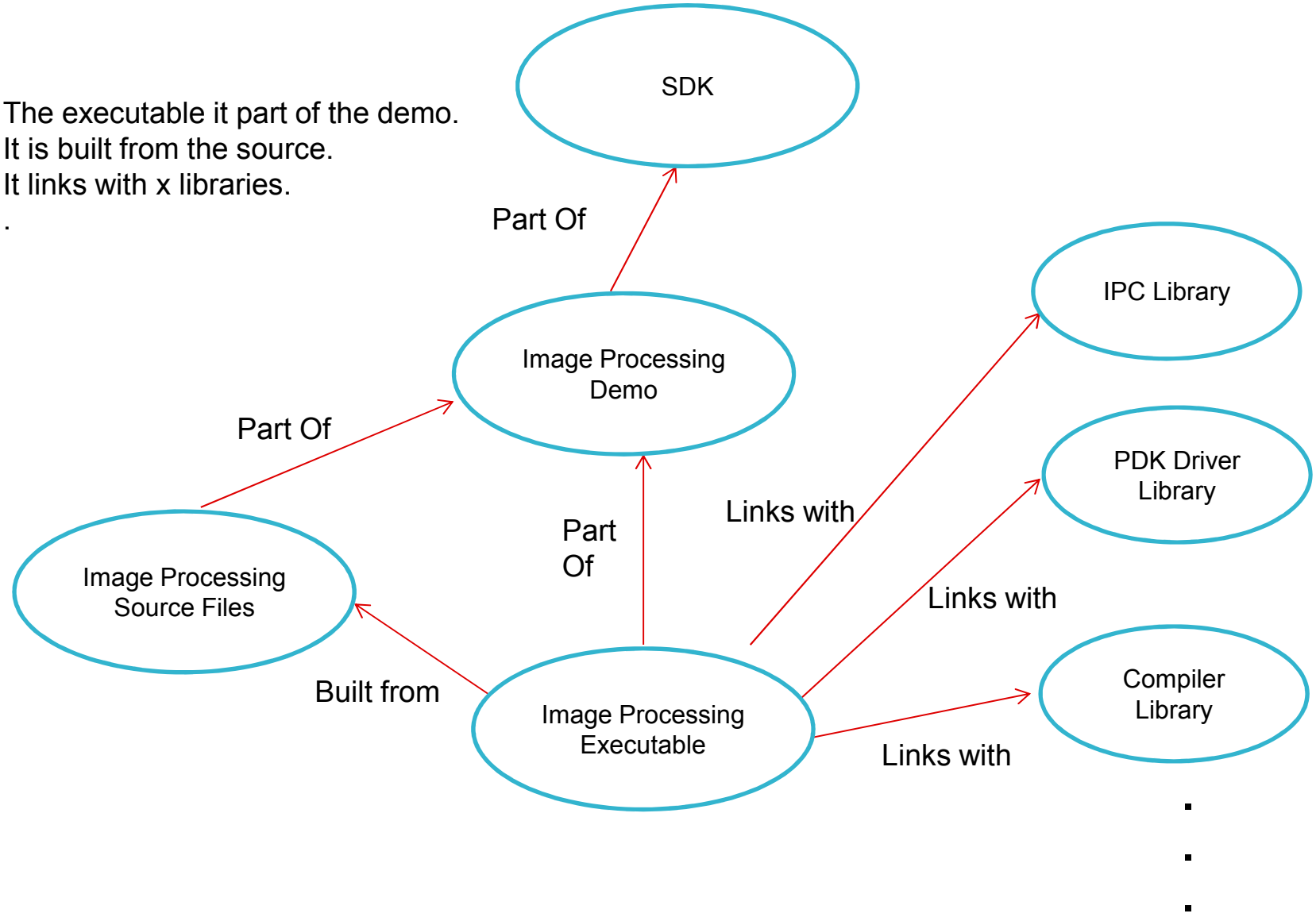
Would like to express the following relationships for the Image Processing Demo



The demo is part of the SDK.
The source will be used to build an executable.
The project files and documentation are used to build the source but are not part of the executable when built.
The bitmaps are outside the executable and are example images that can be used (via an upload to the application).
All of these are part of the demo.

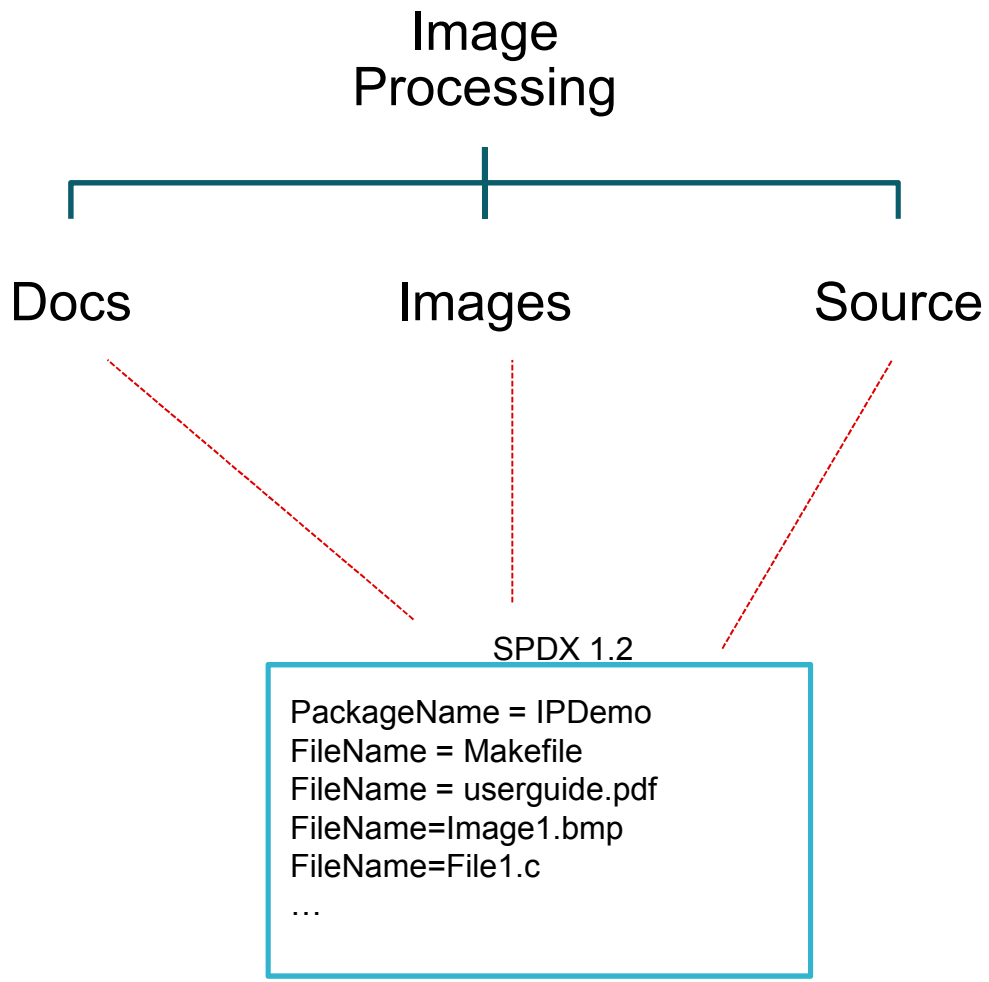
Would like to express the following relationships for the Image Processing Demo

The executable it part of the demo.
It is built from the source.
It links with x libraries.
.

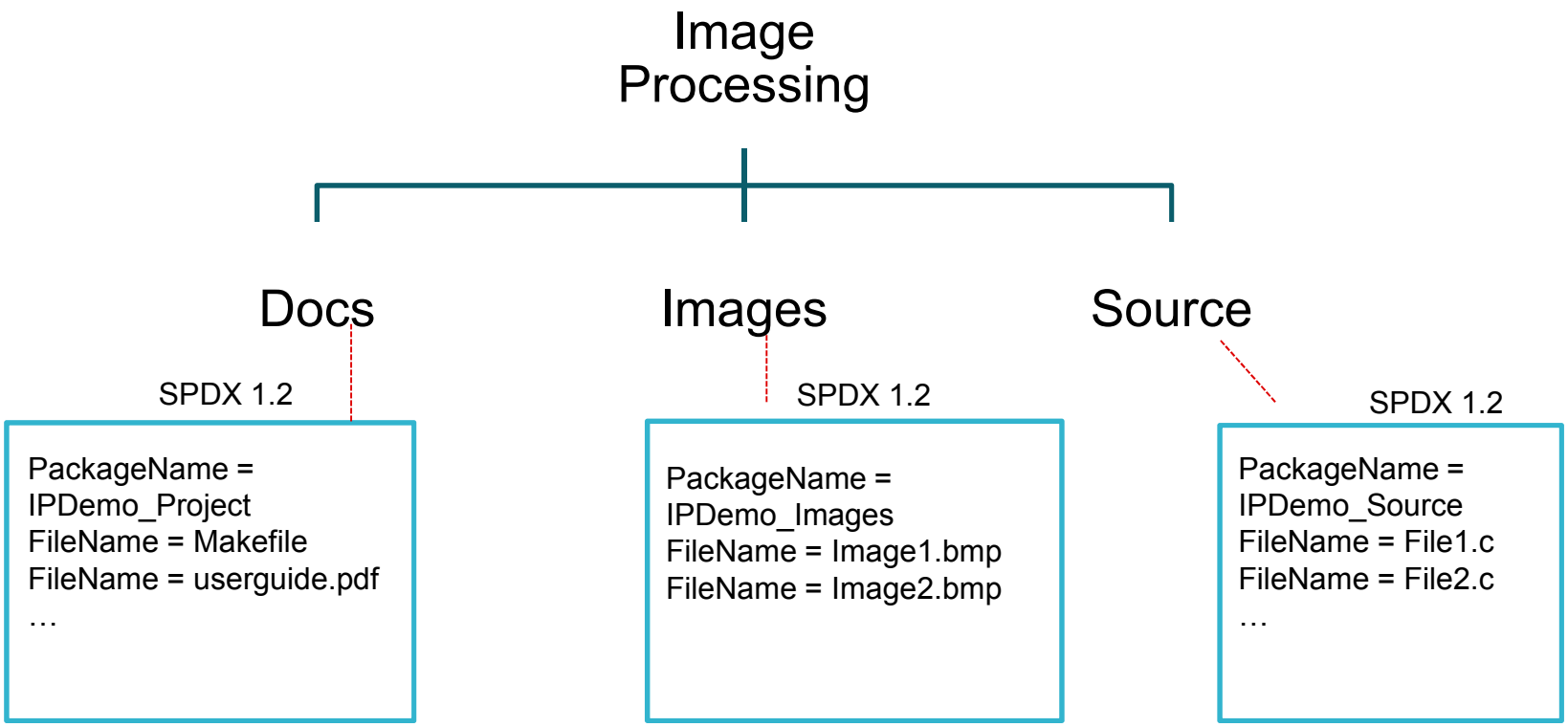


SDX 1.2 and 2.0 Diagrams and Analysis

SPDX 1.2 Instance Diagram for Image Processing Demo – Single SPDX



SPDX 1.2 Instance Diagram for Image Processing Demo – Multiple SPDX

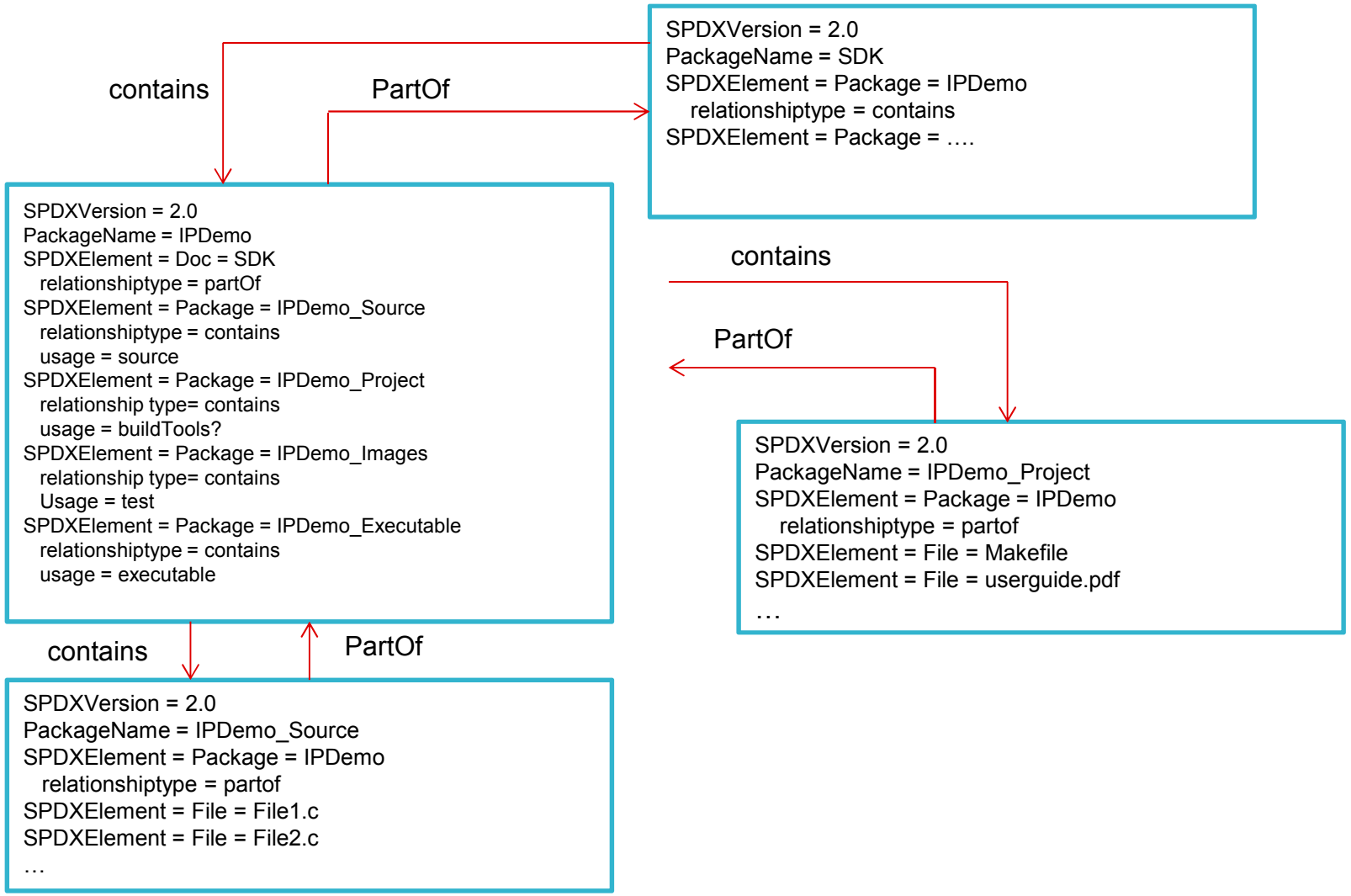


Basically we are breaking out SPDX documents by how the files are used / licensed. No way to relate these package as a single entity. Could just put them all in the same archive.

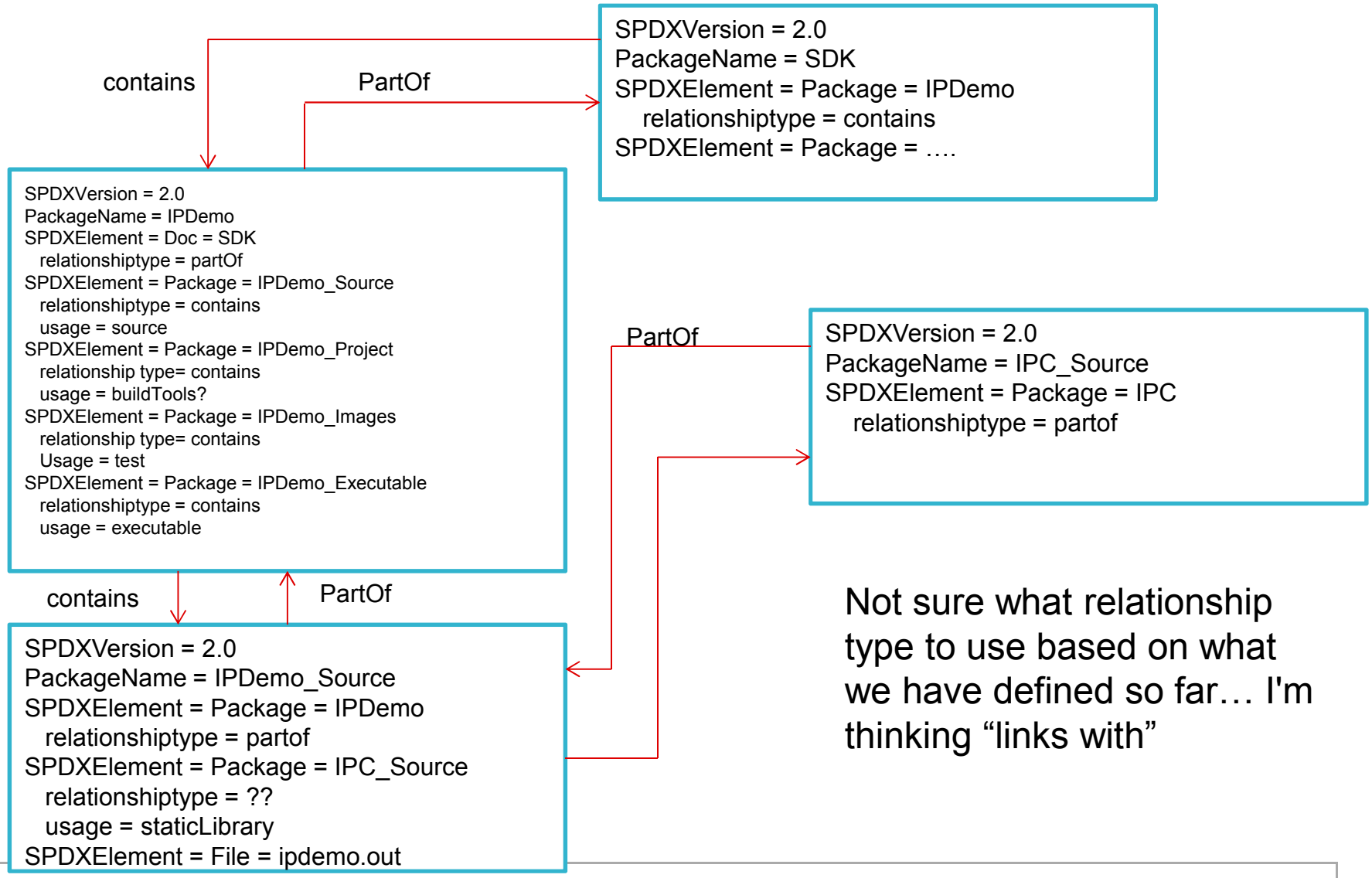
Summary of Analysis of SPDX 1.2

- 1.2 Has no way to express relationships so you are unable to say things like this demo is part of this sdk.
  - Might be possible to convey the same thing by saying putting the demo SPDX file in a tarball of “SPDX’s” for the SDK but it would not be part of the spec.
- Breaking out the different portions of the Image Processing Demo (i.e. source, project, etc) was an arbitrary decision. As shown it could have been one SPDX document but the overall licensing of the bits and how they are used is easier to get your head around if broken out.
  - Breaking it out requires more work.

SPDX 2.0 Instance Diagram for Image Processing Demo



SPDX 2.0 Instance Diagram for Image Processing Demo adding Executable



Not sure what relationship type to use based on what we have defined so far… I'm thinking "links with"

SPDX 2.0 Instance Diagram for Host Tool

We can whiteboard these if needed.
This is fairly straightforward.

SPDX 2.0 Summary of Analysis

- This seems to capture the relationships we wanted to express on slides 9 and 10.
- There would seem to be 3 ways this could be done.
  - A document, package and file all can have a element and relationship.
  - The flexibility is good. Might be confusing when to select Doc vs. Package?
  - Might want to clarify Doc is not the Doc section but an actual SPDX document (need the name of document ads an attribute?).
- Should relationships be circular. That is if A is partof B then should B contain A or is it optional?
- We need definitions for Usage and Relationship values
  - E.g. Does buildTools usage include things like Makefiles and other project build configs (i.e. org.evclipse.cdt.*) or is meant for things like compilers, translators, etc.,.
  - Likely we need to vet lots of examples to make sure we have the right values
  - Will we will allow them to be ”extended”