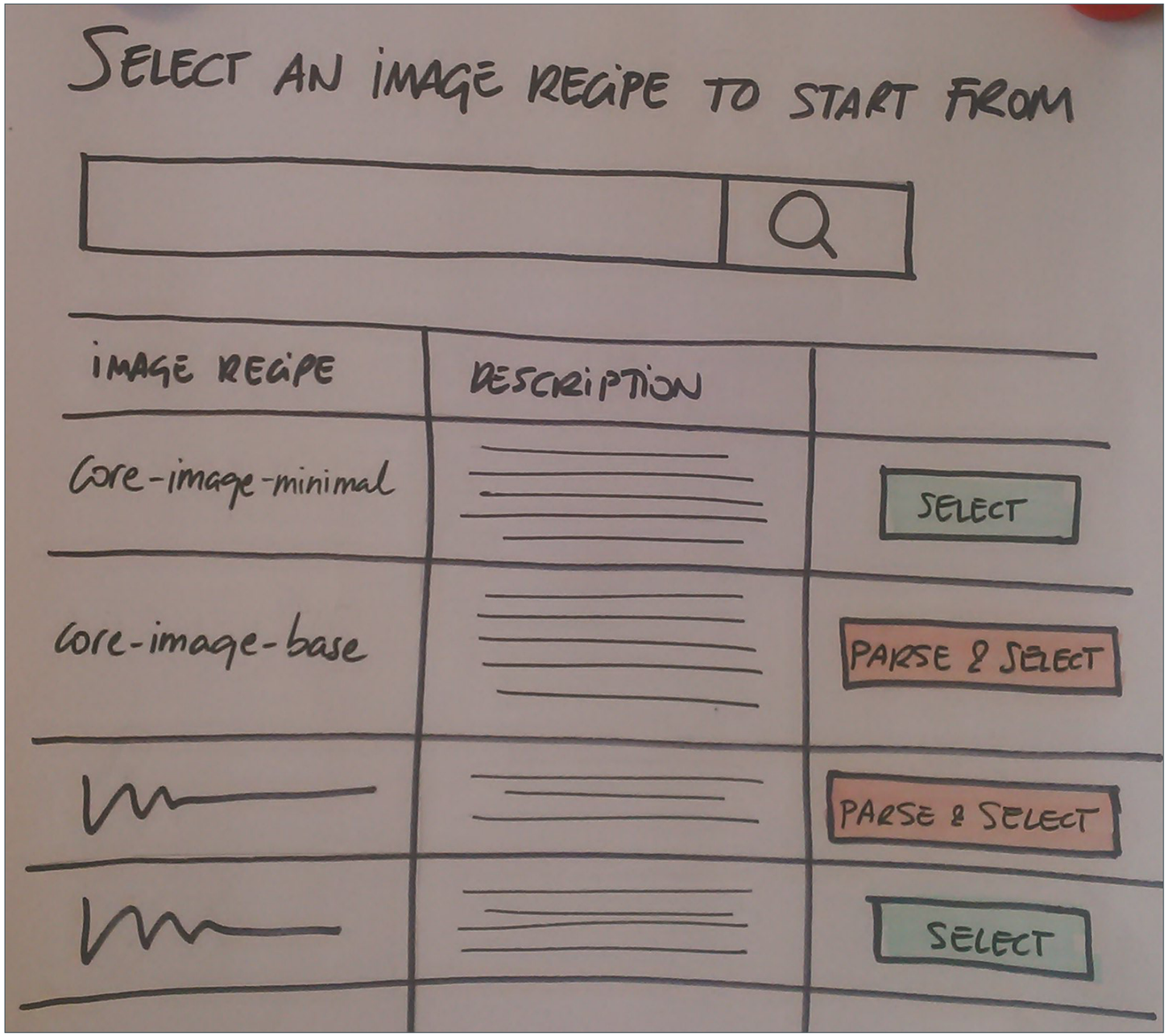
Image customisation in Toaster

High level design



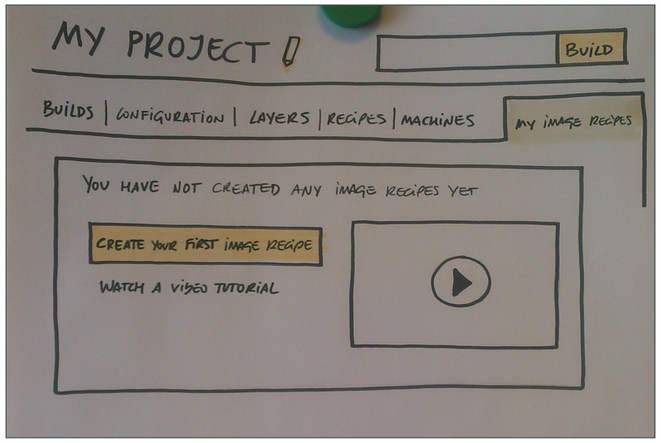
What is this document

This document is the outcome of preliminary design discussions about how to handle image customisation in Toaster.

By image customisation we mean the ability to create a customised image recipe by specifying the list of packages it will install.

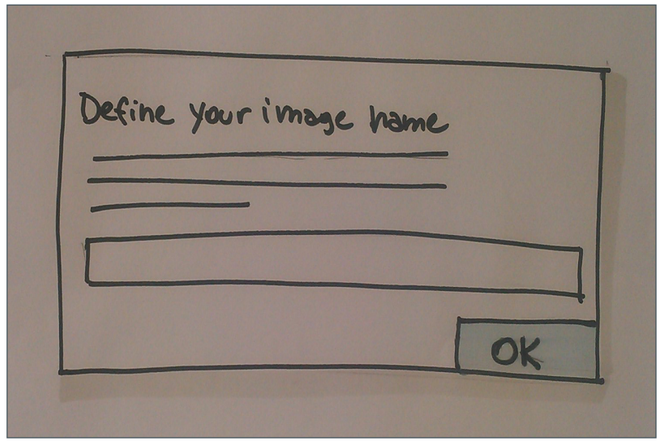
The discussions aimed to provide the following:

- A possible high level design for the image customisation web interface
- A general idea of how such web interface could be implemented during the 1.9 release of the Yocto Project



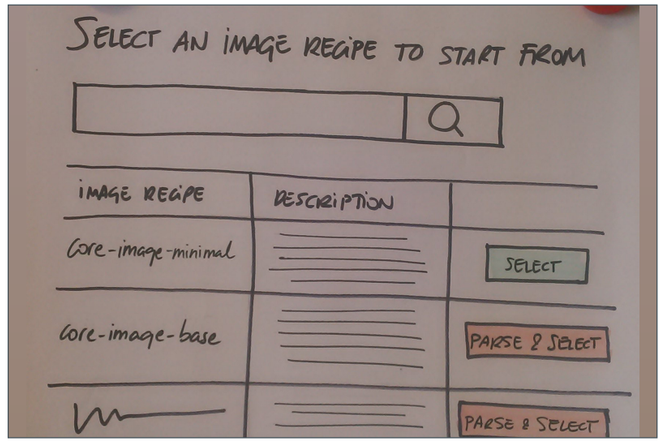
1. The "My image recipes" section

In your Toaster projects, you will find a "My image recipes" section, where you will be able to create, build and manage your customised image recipes.



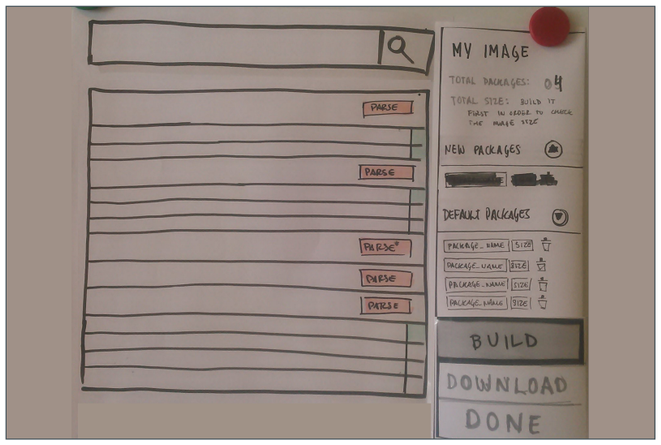
2. Give your image a name

The first step when creating your custom image recipe is giving it a name. Once you do that, Toaster saves your new image recipe. This means you don't need to finish your image recipe in a single go: you can drop it and go back to it at any time.



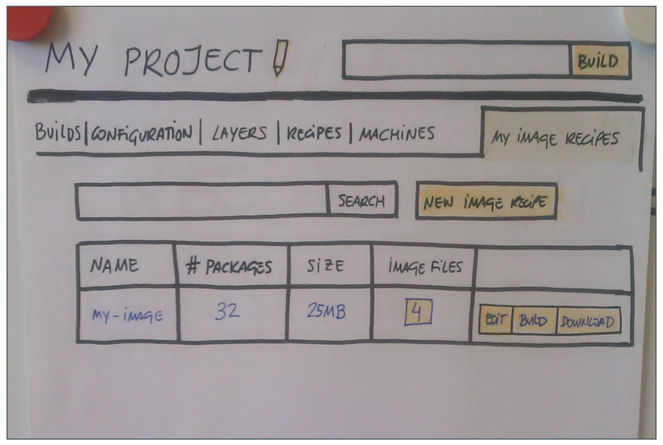
3. Select your reference image recipe

You don't start your image recipes from scratch: that's just too hard. Instead, you select an existing image recipe that serves as a starting point. This image recipe will provide a list of default packages, that you can them add to (or remove from) as needed.



4. Add / remove packages

Toaster will show you all the packages it knows about, so that you can select the ones you want to add to your image recipe. You can also remove packages from the default list. Once you are done, you can build your image recipe, or download the .bb file to integrate it into a custom layer.



5. Once you are done

Your image recipe becomes available in the "My image recipes" section. You can build it, download root file system files, download the .bb file or edit its installed package list at any time.

The improvements compared to Hob

Hob forced users through a once-off, linear customisation process that involved 2 steps: selecting first a set of recipes, then a set of packages produced by the selected recipes. This 2-stage customisation also involved a lot of waiting:

1. For layers to be parsed before you could select recipes

2. For selected recipes to be built before you could select packages

The workflow outlined in the previous page uses the information provided by layer sources and the project build history to eliminate the recipe selection step, and replaces Hob's "modal" parsing with a background process that allows users to complete the image customisation on the basis of the information available. Users can also choose to wait until parsing completes, and even build the image they select as a starting point if they require more accurate information than the parsing process can provide.

What do we need to implement this web interface

1. We need to be able to tell image recipes apart from software recipes and package groups. We have 2 features opened in Bugzilla to provide this functionality: 7575 (for theToaster work) and 7571 (for changes to the OpenEmbedded Metadata Index).

2. We need to be able to "guess" which packages an image recipe produces before building it. This can be done by parsing the layers added to a Toaster project.

3. We need to be able to list all the packages that have been built within a certain project. This is where Toaster really delivers compared to Hob, since Toaster stores the history of all builds within a project, and therefore knows about all the packages that have been built previously as part of it.

4. We need to be able to remember the state of a custom image recipe I am creating, so that I can return to the customisation process and pick up where I left off.

5. We need to handle the issue of which layer custom image recipes go into.

Of these 5 requirements, the trickiest are 2 and 5.

2. How do we handle the layer parsing process

- Toaster will parse the project layers (not all layers Toaster knows about, only the layers added to the project at the time when the parsing starts)

- The information we display to users during the customisation process can have different degrees of accuracy, depending on the layers that have been parsed and the project build history. Users should be able to parse the project layers, or to build their selected image recipe, if they want a higher degree of information accuracy.

- Users should be able to start a parse from the following pages:

    - The layers section in the project page, when a new layer is added to the project (this includes both layer index layers and imported layers)
    - The layer details page
    - The 'select an image recipe' page in the image customisation process

- We might also consider providing a 'start parse' option from:

    - The 'new project' page
    - The 'import layer' page

- We could also start a layer parsing automatically based on certain events. For example, when a project is created or when a new layer is added to the project. For this approach to be feasible, the layer parsing process must not be modal (i.e. must not prevent me from interacting with Toaster while running), and users must be able to stop it. When a layer parsing is in progress, the pages within the customisation process must include a notification informing users that a layer parse is happening.

5. Which layer custom image recipes go into

We agreed that a viable solution for the 1.9 release should not involve pushing to any Git repositories, and so we concluded:

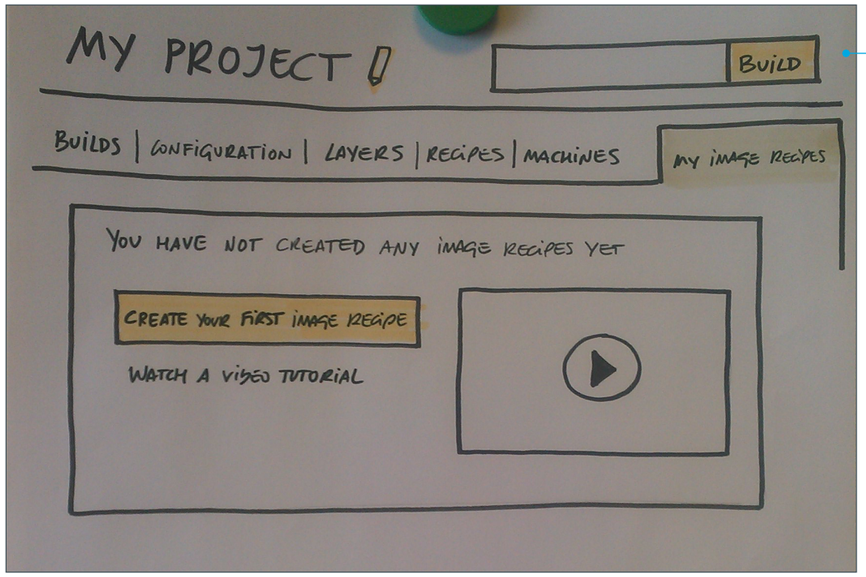
- That the easiest way to handle this is keeping the image recipes separate from the normal project builds, at least in appearance.

- The above means that each project should have a "My image recipes" section, separate from where the normal project builds are configured and launched. You will only be able to build your custom image recipes from this "My image recipes" section, and your custom image recipes will not appear as suggestions in the autocomplete functionality of our build forms.

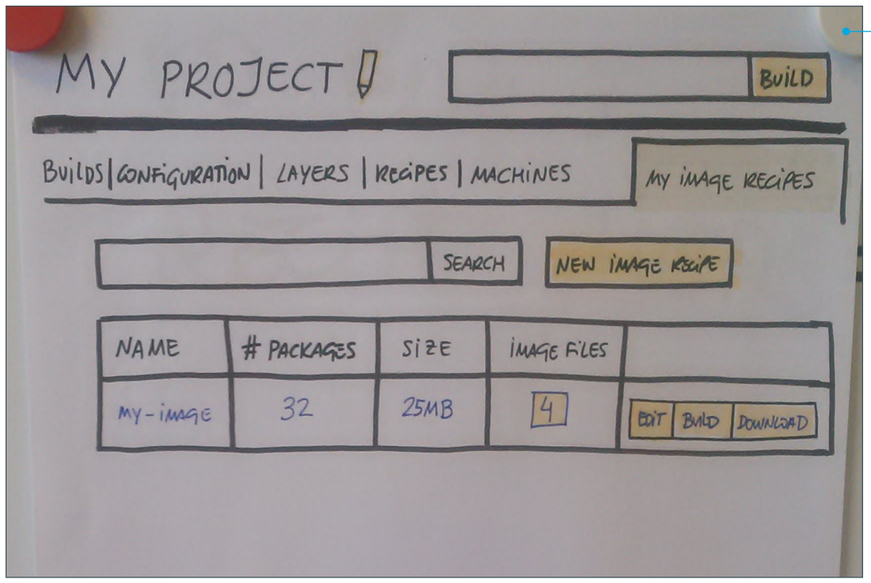
- If you want to integrate a custom image recipe with the normal project builds, you will need to download the custom image recipe bb file (we will provide a download option), add the file to a custom layer, and import the layer into Toaster

- Custom image recipe builds will still show in the Toaster build history (in both the 'all builds' page and the 'project builds' page).

The following pages list some of the design requirements for the customisation interface.



We need an empty state



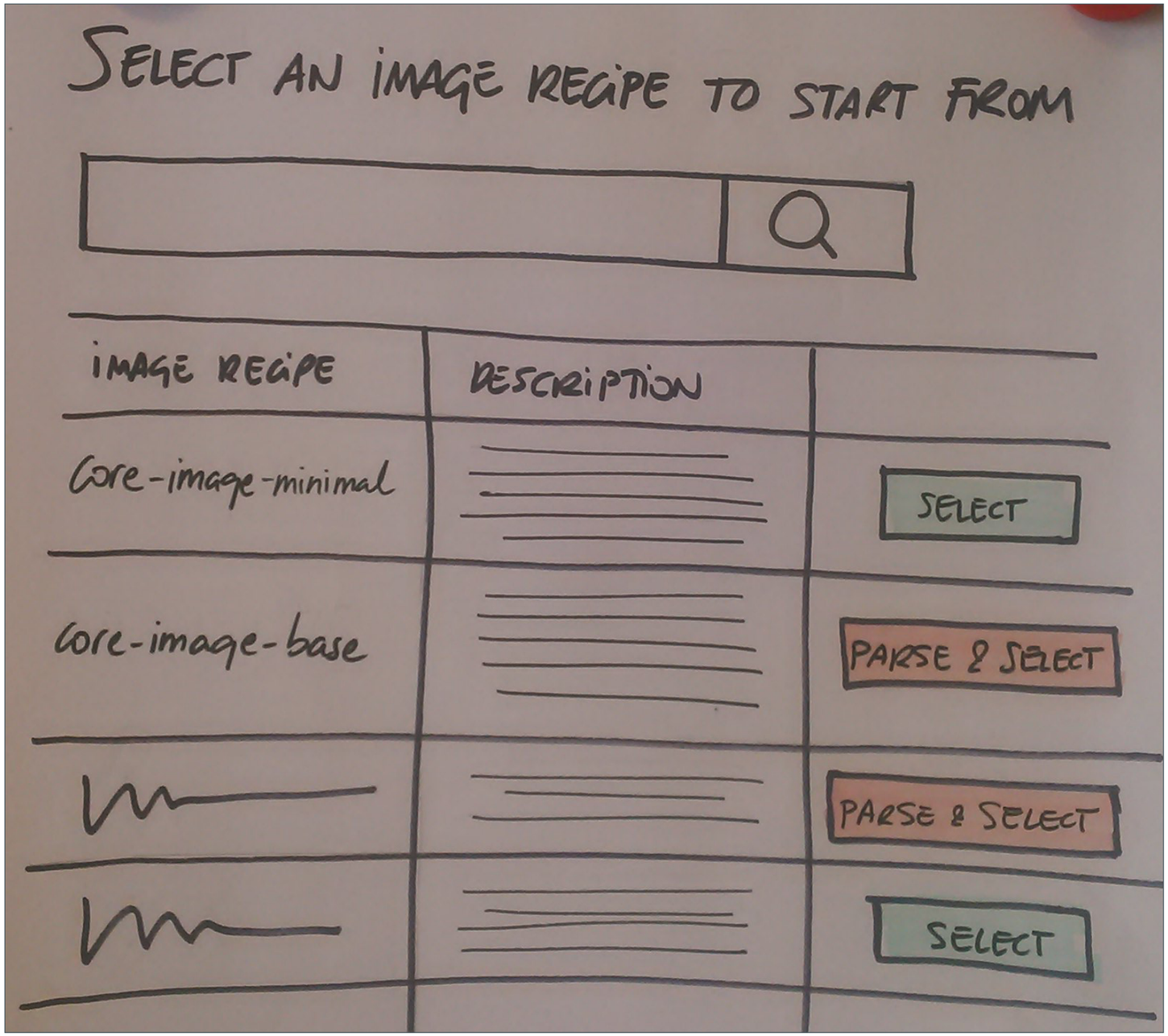
And a populated state

1. 'My image recipes' section in the project page

- We need 2 states for this page: an 'empty' one (when no custom image recipes exist for the project), and a 'populated' one (when one or more image recipes exist for the project)
- We need to provide access to the build information from this section. The information coming from custom image recipe builds will be presented in pretty much the same way as any other build, and will be listed with all other builds in the 'all builds' page and the 'project builds' page.
- The information about these builds will need to indicate if any additional packages where installed due to dependencies Toaster didn't know about during the customisation process. This should probably show as some kind of notification in the image dashboard, and in the table of packages installed.

2. 'Give your image recipe a name' dialog

- This step is modal.
- Once you have given your image recipe a name, we save it in the 'my image recipes' section, so you can leave the process and return to it at any time.



3. Select your reference image recipe

The information in this page can have different degrees of accuracy:

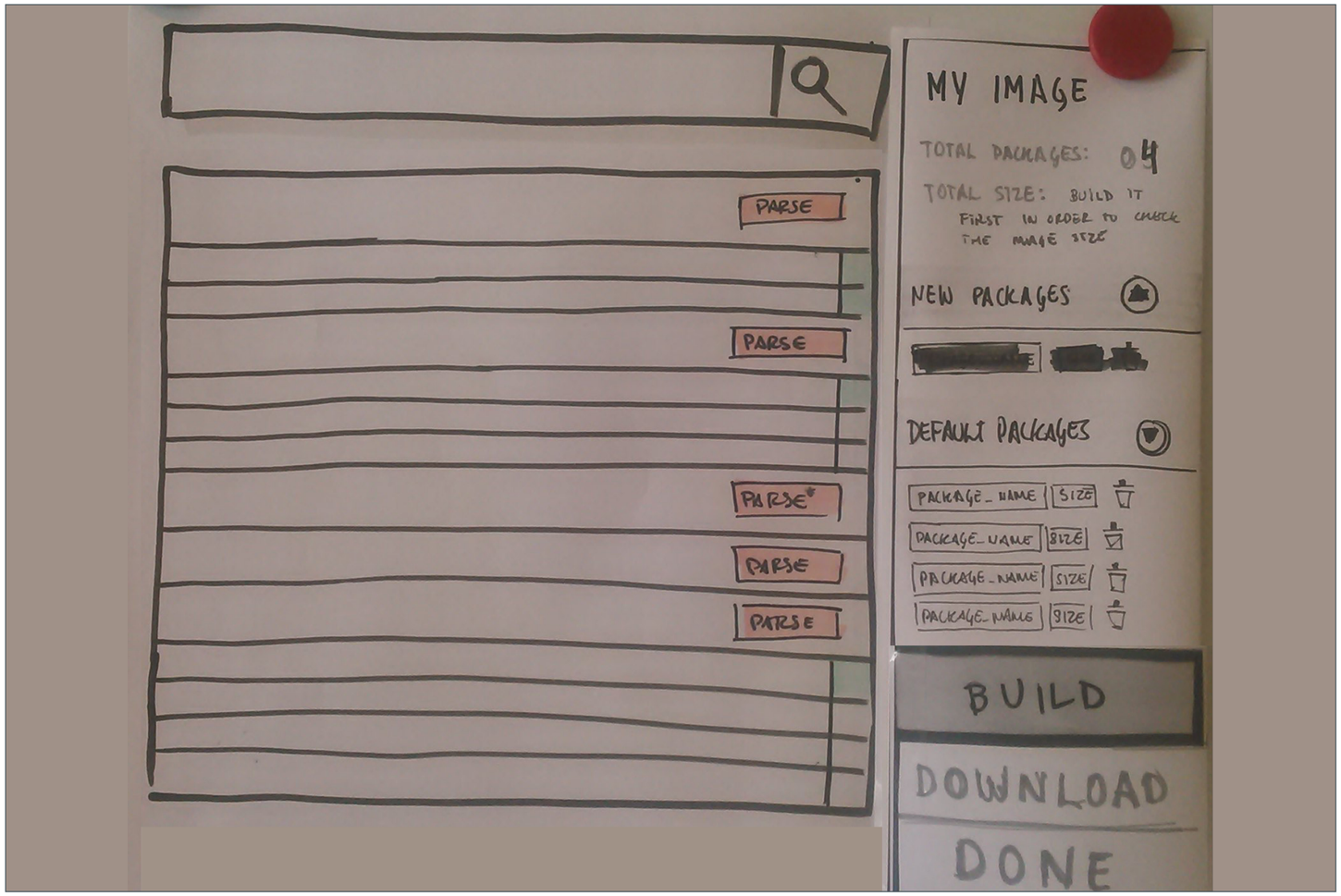
- The page could be completely empty. This is the worst case scenario. As far as I can tell, it can only happen if Toaster does not have any layer sources configured or, if it does, when the lsupdates command fails to fetch the layer information from the layer sources. If the page is completely empty, we would need to check if there are any imported layers in the project. If there are not, we should have an empty state asking people to either configure a layer source or import a layer, as we currently we do in, for example, the project page. If there are imported layers in the project, we should ask users to run a "first-time" parse.

- The page only has information coming from layer source data (no information coming from parsing or builds). When the layer index is set up as a layer source, Toaster should be able to show a list of image recipes to select based on the information coming from the layer index. Information such as number of packages installed and total package size will be empty, and as above users should be asked to run a "first-time" parse before proceeding, since without parsing the layers we will know nothing about the packages installed by any of the image recipes listed.

- The page has information from a layer parse (but no information coming from builds). In this case, we will have an estimated number of packages installed, but no total package size. Users should be able to select one of the image recipes listed and proceed based on the "estimated" information Toaster has.

- The page has information coming from builds: for those image recipes that have been built within the project, we will have complete information, including the number of packages installed and the total package size.

Users should be able to start a layer parse from this page at any time.



4. Add / remove packages

This page is basically a list of all the packages Toaster knows about for the project you are working on. Users will be able to add / remove packages from this page.

We will also need a 'free text' option, where users can type the name of a package Toaster might not know about. The example that came up during the design discussions was optional kernel modules, for which we can only verify that the free text pattern matches what's produced by a recipe (PACKAGES_DYNAMIC). I have no idea what this means, by the way: just quoting Paul Eggleton here.

The information in this page can also have different degrees of accuracy.

- Information based on layer parse: the number of packages installed is estimated, and the package sizes will be missing. When the page is in this state, users should be able to build the image recipe at any point to get more accurate information. The build process should not be modal (i.e. should not stop me from interacting with Toaster or continuing the image customisation process based on the available information).

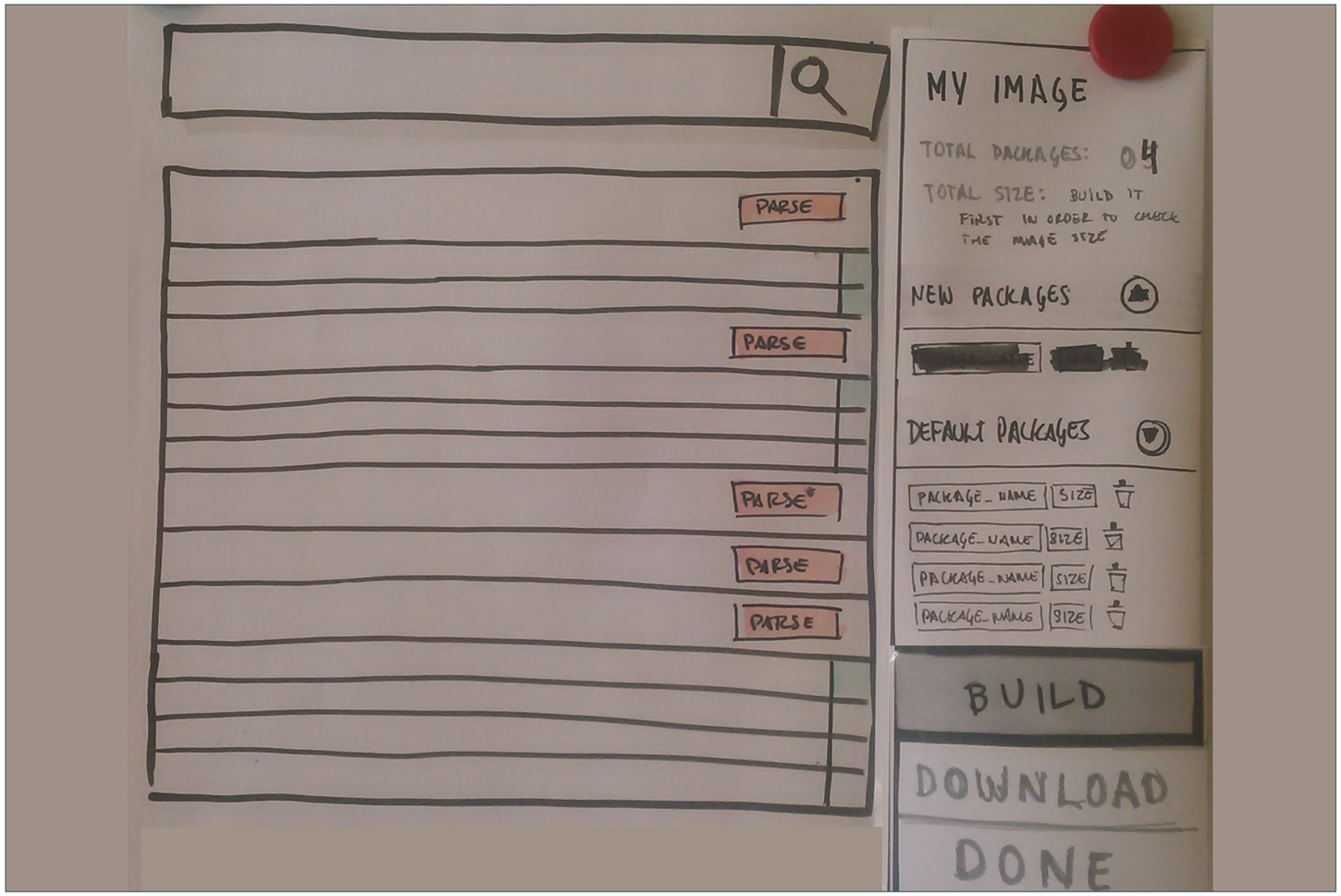
- Information based on builds: we will have an accurate number of packages installed and we will be able to show package sizes.

We discussed the possibility of providing a way of setting the version of a custom image recipe in this page.

We discussed removing any reference to the image recipe selected as the starting point after selection takes place. However, Paul Eggleton has brought up that image recipes might include certain options beyond the package list that users or vendors might be keen on keeping. This means that there are 2 possible ways of creating the custom image recipes:

1. By inheriting the starting point image recipe, and keeping those extra options

2. By not inheriting the starting point image recipe



In general, Yocto Project's core images will use the option 2, but this is potentially something we could turn into an option users could select. Although we should proceed with caution and provide a sane default, since we cannot assume users will know or understand the difference between both, and we are not planning to expose those "options" that will be kept by inheriting the starting point image recipe.

Handling package dependencies

This page will need to handle dependencies when adding / removing packages. We should probably reuse the design we have for handling layer dependencies.

Package dependencies might be compulsory or optional.

Compulsory package dependencies are shown just for information purposes (you either proceed or not).

Optional package dependencies list the packages associated with a check box (like layer dependencies do), that users can uncheck.

All dependencies when adding a package are compulsory. Removing is more complicated. In a situation where package A depends on package B, which depends on package C; and package D also depends on package C

A > B > C
C > D

- If I remove B, the dependency is compulsory (A must go).
- If I remove A, I get the option to remove B, which should be selected by default.

The problem with this approach is that, if I deselect the option to remove B, that might change the list of packages shown as being removed. If this proves too complex, we'll make all dependencies compulsory.

Additional features

A few additional features came up during the discussions, but we agreed we will not be able to deliver them in the 1.9 release:

- Breaking up package groups. We have not tackled this issue yet, mainly because we don't think we can get to it as part of the first version of the image customisation functionality.

- Breaking package dependencies via configuration options. If I remember correctly, this is about breaking RRECOMMENDS, but must be done for the whole image recipe (cannot be done on a per package basis).

- Enhance layer removal. Currently, Toaster does not use layer dependency information when you delete a layer from the project: you must remove layers one by one. We could enhance layer removal by asking users if they want to remove the dependent layers if no other project layers depend on them.

A question we forgot to ask

What happens to my custom image recipes if I change the project release?