GDP viewed as a portfolio

General approach
1.- Segmenting our target
2.- Characterise the new target groups.
   2.1.- Additional considerations
3.- Defining GDP outputs for each target
   3.1.- Target group -> output
   3.2.- Outcome -> target group
   3.3.- Where are we?
4.- Transition from one output to another
   4.1.- Moving towards master
   4.2.- Moving away from master
   4.3.- "Keeping the distance" from master.

• Author: Agustin Benito Bethencourt. GDP Delivery Team Lead. Codethink Ltd.
• Mail: agustin.benito@codethink.co.uk
• Date: 06/06/2016

This is an attempt to have a structured discussion about what the future of GDP might look like.

Why now?

Many of the activities we need to do in the coming future will be better understood if we share where are we heading to. The discussion is useful even if we don't reach a consensus any time soon. At least we will know where we dissent.

GDP target developers. I believe it is positive to clarify what do we understand by "developers" moving down from an executive to a portfolio management level, it is still a too wide target group.
General approach

If we are able to segment "developers" we will be able to define an output for each target (sub)group. Once the output is defined, we can start thinking about how the transit path from one output to another one should look like, considering our ultimate goal. This way we can take actions to reduce the effort required to steer any segment towards the most desired output for them and for the overall project.

How do we fit the above into GENIVI's calendar, the relation between GDP and Baseline and the transition from where we are to the final picture are out of the scope.

So my goal is: 1.- Segment our target "developers" -> 2.- Characterise each segment (target) group -> 3.- Define an output per target group -> 4.- Design the transition from one output to the other.

1.- Segmenting our target

In the current GDP context, "developers" can be segmented into three main groups:

- **Group A**: those engineers who work on code that GDP already ships, specially those who work on key components. GDP is not a "platform" for these engineers but "the subject" of their work.
- **Group B**: engineers that uses GDP to develop additional software, not included in GDP. For them, GDP is a "platform" to...
  - **B.1** contribute to GDP new features. These engineers become part of Group A after their software is merged.
  - **B.2** develop software that is not meant to be included in GDP, but is in direct relation/dependency with what GDP delivers.
- **Group C**: engineers who uses GDP for other activities or develop software with a weak dependency with what GDP delivers. We can consider GDP as a "commodity" for them. Maybe a platform in some contexts.
2.- Characterise the new target groups.

The next step is naming each group/subgroup of "developers”.

- **Group A:** GDP hackers
- **Group B:**
  - Group B.1.: GDP power users
    
    In GDP context, these are the users that have knowledge about GDP and are willing to test, report bugs and provide basic support to other users.
  
  - Group B.2 and Group C: GDP users
    
    Developers from Group B.2 will have knowledge about specific components only. They generally limit their participation in the project to support request.

- Group C is formed by developers that uses GDP but does not understand how it works or does not participate in any area of the project. They can be considered as GDP consumers. I propose to refer to them also as users, for simplicity.

2.1.- Additional considerations

- I use the word "output" in this explanation to avoid the word "product" since that word is reserved to commercial environments. I do not like to use the word "project", which is the standard one in Open Source environments. Suggestions in this regard are more than welcome.
- For this discussion, group B.2 and C are so alike that GDP project will deliver the same output for them.
- I understand that in some environments the usage of the word "hacker" might seem strange since media has assigned it a negative connotation. That is not the case in Open Source technical environments, even in the distro world. In fact, most Open Source engineers would welcome any effort we can do to revert the current situation.
3.- Defining GDP outputs for each target

3.1.- Target group -> output

- GDP hackers: **Master** is the output for this group. We expect them to build GDP themselves with what they need/want. Master should always work for them. Ideally we can reach a state in which we can properly talk about a rolling delivery.
- GDP Power Users: those who want to contribute code need to be up to date with master but first they need a good base, easy to start with, that works. So, my expectation is that they download the latest available **image** (release). Later on they move into master to test their new developed software, specially if they want to make it part of GDP. once it is part of GDP, they go back to consuming images as default output.
- Users: **images** are the output for this group.

If we decrease granularity in our target segmentation in the future, we can consider additional outputs like an "untested" master or a GDP SDK.

3.2.- Outcome -> target group

Looking at this from the opposite angle:

- 3.2.1.- A **rolling output** with the latest tested software, always available to be built, for GDP Hackers and Power Users. Since bringing innovation to GDP is our main executive goal, we need to consider as measurable success criteria in a first stage to increase the number of "Master consumers", so GDP hackers. So Master is our main output.
- 3.2.2.- **Minor releases** in a fixed shorter cadence for Power Users as target, reducing the gap between major releases and master. Based on the behaviour of our current power users, our resources... I predict simple/low effort minor releases at the beginning and more complex later on.
- 3.2.3.- **Major releases** in a predefined cadence, making very easy for Users to consume GDP.
3.3.- Where are we?

As we are still in the early stages of the project, it has been vital that we strive to increase the number of contributions and active contributors. In order to do that, we need to increase our user base. To achieve that goal we required to make GDP easy to consume so we have focused on making major releases only so far.

We just took the first step towards creating "master".

4.- Transition from one output to another

Having defined target groups and outputs allows us to plan an approach to encourage more developers to become a part of each target group, and further plans as to how these developers might ultimately transition from one output to other. Our release process for each output has to be coordinated to ease some of those transitions. I refer to this as "portfolio release model".

4.1.- Moving towards master

Targeting directly potential GDP hackers is "in general" unrealistic in our context. The more realistic approach is having users becoming GDP hackers after being Power Users. I consider this our "critical path".

Since we are in a professional environment, I suggest to put effort first in making the transition from any minor release to Master as smooth as possible. The best initial approach is considering minor releases only as entry points to master so we only need to clarify how after installing a minor release I can update to Master.

We cannot leave major releases aside, since they are our main "single shoot effort". So we should define the major release process as a minor release + extra processes. For this purpose, Major and minor releases should be the same.

Once this transition is set, the next one should be to transit from a major to a minor release avoiding re-installation.
4.2.- Moving away from master

This path does not make sense in our context so I do not see GENIVI supporting it anytime soon. A different story is if Baseline and GDP are both born from "Master". This was the main idea of my talk at 14th AMM.

4.3.- “Keeping the distance" from master.

Since we are not targeting users as priority one, the transition from one major release to the next one will be out of scope for now. Considering minor releases as entry points to Master means that transitions between minor releases will not be supported either in our current context.

In summary, I do not recommend discussing about releases and targets without structuring the conversation, providing a portfolio projection. What we have today are pieces of a puzzle. They might need to be fixed in order to fit, but there are additional ones.

That's all for now. I will provide more detail about some of these ideas in the near future if they can fly.

Agustin Benito Bethencourt
Codethink Ltd