"The greatest challenge to any thinker is stating the problem in a way that will allow a solution."
Bertrand Russell

Targets: The Users
v. 0.2

Mihnea Căpraru

Purpose

This document is meant to define a strategy for defining our goals. I do not claim nor intend to define this strategy all by myself, this is only meant as a call for further improvements.
When I say 'our goals', I mean, as the title says, our goals while marketing KDE to software users proper. Marketing to enterprises, organizations, government or software distributors are different problems.

The Question

What categories of users do we want to persuade to use KDE?
Related to this: what categories of users that we already have do we want to keep/make more loyal/satisfy better?

How to Answer the Question

We should first answer two preliminary and mutually independent questions:

A. What categories of users do we wish that we had?
B. What categories of users can we really have?

These questions are to be answered by indicating two sets of user categories. The answer to our main question will then easily be obtained by intersecting the two sets.

In order to be able to perform this intersection, we need to answer both Question A and Question B with respect to the same universe of discourse.

What this means:

The 'universe of discourse' comprises all categories we can speak about.

Example: U = {apples, pears, plums, bananas, cherries}

The answers to A and B are both subsets of U.

Example:
“We wish we had {apples, bananas, cherries}”
“We can have {apples, pears, cherries }”

And the conclusion is the intersection of A and B.
“We should target apples and cherries.”

Defining the Universe of Discourse

Preliminary.

We are free to identify how many categories of users we like, but our purpose is to identify them in a way that suits our purpose (providing reliable answers to A and B). This means that the Universe of Discourse suggested below is not to be taken as merely a matter of convention. Getting it right is very critical to our success, it means as much as using the right conceptual tools.
Removing redundant 'attributes' (this is a forward reference to a term) and adding relevant ones is therefore very important, so please track my mistakes and hunt them down.

Logical Approach

From ancient times and up to the nineteenth century, many philosophers have hedged the bad habit of trying to use porphyrian trees in order to account for ontology.
Don't worry, what I've said has a meaning. To make it clear:
A porphyrian tree is an order-species tree. Think of the way biologists classify living beings: we can have plants or animals, animals can be vertebrate or non-vertebrate, vertebrates can be fish, amphibians, reptiles, birds or mammals etc etc.

The Java class hierarchy is a porphyrian tree too: everything is an Object. Objects are divided in all known subclasses of Object etc etc.
A file system is a porphyrian tree.
(Actually the UNIX file systems provide hard links, so they are not really porphyrian, but that's an irrelevant detail.)
KControl's tree widget is a porphyrian tree.

I hope these examples have cleared up the advantages and disadvantages of porphyrian trees: while they have a simple structure that makes them excellent tools when reality is simple, they are fatally limited and become arbitrary when reality is complex.
As for 'ontology', this means the part of philosophy that tries to tell us what is real and what not. What kind of things there are.
We are not interested in philosophical ontology here, but we are interested in a specific problem: what kind of users are there?

The user landscape is too complex for a tree to handle, we need a more flexible tool. This is why I suggest that we use attributes (or maybe call them 'tags', or 'predicates').

This means that we need to identify the relevant questions that we can ask about a user, and provide a domain of answers for each question. Then we multiply all possible answers (that is, we take one for each question) , throw away the impossible and the very rare combinations, and we have our universe of discourse.

(Actually, it would probably make sense to also throw away the combinations that are neither impossible nor very rare, but still rare enough to not be an attractive target).

Example (this example uses apples, not users) :
Tag 1. Color
Question: What color does the apple have?
Domain: {green, red, yellow}
Tag 2. Age
Question: How old is this apple?
Domain: {less than one month, 1-3 months, 3-6 months, more than six months}

Multiplying the tags:

First we provide the cartesian product of our domains:
{ (green, less than on month),
  (green, 1-3 months),
  (green, 3-6 months),
  (green, more than six months),
  (red, less than on month),
  (red, 1-3 months),
  (red, 3-6 months),
  (red, more than six months),
  (yellow, less than on month),
  (yellow, 1-3 months),

(yellow, 3-6 months),
(yellow, more than six months)
}

Now we sort out the impossible/unlikely combinations (young and only young apples are green), thus we obtain our universe of discourse:
{ (green, less than on month),
(green, 1-3 months),
(red, 1-3 months),
(red, 3-6 months),
(red, more than six months),
(yellow, 1-3 months),
(yellow, 3-6 months),
(yellow, more than six months)
}

Useful Tags

What tags can we put on users? Which ones of the vast variety of them are the ones that would help us frame our problem in a solvable matter?

This is what we need to decide, here's a sketch:

Tag: Main Place of Use
Domain: home, work, school, kiosk

Tag: Secondary Place of Use
Domain: home, work, school, kiosk, none

Tag: Subjective Stance Towards KDE
Values: loyal user, likes KDE, indifferent, dislikes KDE, hates KDE, never used KDE

Tag: Main Use of Computer
Values: office work, leisure, software development, study, arts, science, web development

Tag: Technical Knowledge
Values: expert, geek, average, newbie, grandma

Tag: Desire to Learn About New Things Computer Related
Values: high, low, not at all (aversion)

Tag: Choice
Values: user chooses software, user doesn't choose

Tag: Authority
Values: authority to 'buy', authority to suggest buying, no authority

(Note. Choice and Authority are somehow overlapping, but they are not identical. People often use what's provided by someone else, although they do have the authority to choose for themselves)

Tag: Coder
Values: yes, only simple scripting, no

Tag: Artist
Values: yes, no

Tag: Knows/cares about FOSS
Values: both, neither, knows but doesn't care

Tag: Tastes Concerning Looks
Values: elegant, flashy, corporate

Tag: Favorite Color
Values: blue, green, red etc etc

(Really this is very important – only supporting one color (example blue or brown) means dropping many potential users who have strong feelings about colors)

Tag: Dependency on Non-Portable Solutions (like Access applications)
Values: insurmountable, high, surmountable, low/none

Tag: Speed Freak
Values: performance is crucial, performance is important, performance only matters when the applications really don't work

Tag: Desire for Unity
Values: integration highly valued, integration appreciated, indifferent, likes skinned apps/other inconsistencies

I'm quite aware that some more relevant tags can be found. Also the domains can be defined better.

We assign two weights to each tag to account for its relevance: one for desirability, one for feasibility.

Example:

Tag: Coder

Desirability weight: 1.0 (whether somebody is a coder or not matters a great lot to us)
Feasibility weight: 0.7 (whether somebody is a coder or not affects our chances to lure him/her into KDE significantly, but not 'a great lot', since coders are also end users, after all)

Disclaimer: the above is an example, not a thesis.

After we have a final list of tags and their respective domains, we should take them tag by tag and ask our questions A and B for each of them. This will analyze a daunting, complex problem into several minute, relatively easy to solve problems.

We assign a weight between 0 and 1 to each value to account for desirability, and one to account for feasibility (that is, how much we would want the user, and how able we are to get the user).

Example:

Tag: Favorite Color
Values: Blue, Red, Green, Brown, Yellow, Orange

Desirability: .5, .5, .5, .5, .5 , .5

(this means we don't care what the favorite color of the user is)

Feasibility: 1, .5, .5, .5, .5, .5

(this means that it's very easy to market KDE to someone who likes blue – all other things equal – but it's more difficult to promote it to the other guys, since KDE can only be made to imperfectly have another color through significant customization – such as find green-friendly icon theme, never mind the 'About' screens etc)

The above is just an example, I hope it's illustrative.

After having reliable answers to all tag-specific problems, we can easily synthesize our final answer. This is how we do it:

Assume we have n tags, t[0], t[1], ... t[n-1]

For each tag we have a domain with m values: v[0], v[1], ... v[m-1] (m is n-specific, i. e. different tags can have different numbers of acceptable values)

Each possible walk from top to bottom through the list of tags gives us a potential target user profile

Example path:

home, work, like it, newbie, low, authority to buy, no, no, both, elegant, red, low/none, no speed freak

For each path we determine the desirability and the feasibility, thus:

we multiply the desirability weight of each tag with the desirability of the value that our path includes for this particular tag

Example: suppose that the desirability weight of the 'Coder' tag is 0.8 (i.e., whether someone is a coder or not affects our choice quite a bit). Suppose our path has 'no' (we are speaking about a non-coding guy). Suppose the value 'no' has desirability weight 0.3 (less than average, we want non-coders less than we want coders). Multiplication gives 0.8 * 0.3 = 0.24

After we did this at each tag's level, we add the products. 0.24 for the coder tag + , say, 0.45 for the 'technical knowledge' tag + ... = the final desirability sum of our path.

Note: I hope it's not just a piece of mathematical FUD when I'm saying that we are really determining the scalar product between a vector which defines a user type and a vector which defines the relevance of the tags :-)

We do just the same with the feasibility tags, for each path

This may sound scary, but it's really easy computations (we might as well write a small app to help). The more challenging problem is providing the right input, i.e. assigning reasonable weights to tags and values.

We now have lots of user types. Some of them are very desirable, some are relatively easy to reach. We must choose the right ones amongst them.

In order to do this we must identify the user types (paths) which have:

- high desirability
- high feasibility
- and high numbers.

Note: it is very important to think at numbers at this point and not sooner. When we estimate the desirability of the values of a particular tag, we only think at **one** user (and whether we prefer the user to be a scientist, for instance) . We don't bother with estimating how many such users there are. We only take this into account at the end of the process, when we decide whom we're actually targeting.

At the end of the process, we won't have broadly spread answers such as 'we should target newbies'. This is because newbies are not all the same, they can greatly differ with respect to other tags. What we get is a set of paths, i. e. precisely defined user personalities.

At this point I suggest that we define 'personae'. We give each path that will be targeted a proper name (such as “Jack”, “Manuel”, “Inge” etc). This technique is usually employed by usability designers, but it can help us too. Having a set of personae to talk to would greatly help us shape our messages, know whom we're talking to. Besides, if we can promote our personae to the usability team and the developers, we might have a huge winner.

Of course, when I say 'promote to...', I really mean talk to them when assigning the tag/value weights. It would do no good at all to decide to target certain users, if the developers themselves targeted other users – we would only have defined a message inconsistent with the product.