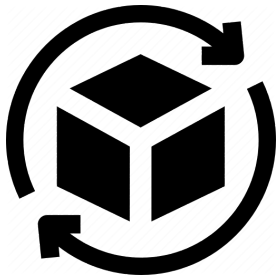
By Tiberius Brastaviceanu, Francois Bergeron add your name here

Sensorica©™

Design Philosophy



under construction...

This document explains the design philosophy within Sensorica,
an Open Value Network.

If you contribute to this doc **make sure you respect** the Content rules

---

NOTE: In some places in this document we are consciously avoiding the use of the term ***product***, to steer clear from *cognitive interference*. For most people, the term ***product*** designates a material good or service that is distributed through markets. Linux, the operating system is not a product. Wikipedia is not a service that people pay to access content. The property regime of Linux and of the content on Wikipedia is common. Solutions built through *commons-based peer production* are not destined for market transactions, and thus they don't fit the traditional economic definition of a product.

Table of contents

Links

Start an open venture (a project)

Sensorica service system - remote monitoring option
Sensorica's sensors - the vision
Sensorica licensing

How to play the open game... (post by Tibi on Multitude Project blog)

Introduction

The philosophy behind our deliverables is not only an ethical issue, it is also a very important economic issue.

See also discussion on Licensing.

Sensorica's mission

Current mission[1]

> *We are committed to the design and deployment of intelligent sensors and sensemaking systems (IoT + AI + p2p infrastructures), which allow our communities to optimize interactions with our physical environment and realize our full human potential.*

Main characteristics of hardware deliverables

NOTE: these characteristics are linked to other OVN processes like manufacturing, outreach, dissemination, service, etc.

- **Open source** - share design and permission for economic exploitation, build communities
- **Shareable** - easy to find, use, transport, facilitates transactions among users, safety and security
- **Modular** (perpetual and customizable) - see this link about a specific effort to define a

[1] if changed please update! see Sensorica mission document

standard for scientific parts.
- **Interoperable** - use shared standards, allow combinations
- **Socializable** - allow sharing of actions made through the intermediary of these deliverables
- **Ecological** - sustainable
- **Ethical**
- **Accessible** - low cost, low assembly skills, local materials

Why open source hardware solutions?

An economic choice:

- User participation - trust and transparency
- Interoperability - more options
- Innovation speed - because many brains
- Fast adoption
- Early adopter integration - customization
- Low development cost
- Multiple application
- Build things with communities behind them for continuity of innovation

The lego lab concept

Open Legolab concept main doc

The labonline concept

Open Labonline concept doc

Some arguments

About perpetual things

Planned obsolescence was implemented because it can be profitable in a market / transactional economy. There are other models that coexist with it, for example the PC (personal computer). According to Moore's law, any configuration of a PC becomes obsolete in only six months. But the modularity of the PC allows the user to update it by replacing internal parts. A PC is something close to what we call a ***perpetual thing***. It can evolve continuously and even morph into something else.

One disadvantage of products planned for obsolescence (apart from the fact that they are not sustainable from an environmental perspective) is that every time the consumer needs to replace the product he can switch to another brand. Companies that implement this practice need to spend energy to keep their customers, to generate repeated sales. Perpetual products build long-term relationships through service for example.

Every time a product becomes obsolete the user has the opportunity to go with another brand, if he/she is not trapped by network effects[2]. Therefore, one way to secure profits when planned obsolescence is applied is either to have a VERY good brand (the perfect example is Apple), or to have that product embedded within an ecosystem of products to create strong network effects (the best example is Microsoft).

A perpetual thing (almost) never becomes obsolete. Only a given configuration can become obsolete because of technological advancement, which is NOT *planned obsolescence*, but rather *natural obsolescence*. In this case, there is no reason to change the thing or to change the brand. If the thing can be easily updated / upgraded the user will just keep it. This helps the producer to establish a longer-term, more profound relationship with the user.

In software, this tendency is very pronounced. For example, Google updates its products on a regular basis without interfering too much with their use by their users. Users get accustomed to a certain environment or interface and tend to stick with it, unless the evolution lags behind other equivalent products available.

Moreover, if this thing is open source, it can generate around it an ecosystem of other things, which in turn generates its own network effects. Arduino is a good example of this.

A perpetual thing is actually economically viable. As for environmental concerns, I would

[2] Referring to the need to have the same brand because of compatibility issues with other products used by the same user or by other individuals with whom the user interacts very often

even say that its future is assured, because apart from economical viability it can also be better branded.

Inherently open
We are designing things that are inherently open, that are modular and can be easily updated, that are NOT programmed for obsolescence or made difficult to modify. This makes them incompatible with closed products (designed with control in mind). In other words, our deliverables cannot be simply copied by classical business organizations, which design on a very different philosophy.

An ecosystem

Modularity and *interoperability* allows the creation of an ecosystem. Moreover, the energy to build this ecosystem is drawn through network affiliations and open source collaboration. The value of each deliverable is reinforced by the others.

See more on How to play the open game... by Tibi.

Modular architecture

[From **Bob**]

1. **Configuring by customer order**, where all of the options are separate products offered for sale, and the configuration is just the selected order items. So for example, in a Mosquito scientific instrument system, you would offer a "base" system containing the common features, and each the variable options (e.g. the transducers) would be listed for sale, and the configuration would be the transducers and other options that were ordered.

Advantages of configuring by customer order:

- It's flexible
- and it's simple from a software development viewpoint. You'll need customer orders anyway.

Disadvantages:

- Can't tell the difference between a product configuration and other random items on a customer order.
- Customers need to know how to configure the product, or need a lot of help in ordering (as in, have somebody who knows how, enter the order for them).

2. **Configurators**, configuration models and configurations: for example, the features and options solution I mentioned in a previous message in this thread.

Advantages:

- The order form guides the customer through the process of configuration. You

have experienced this process if you have ordered a computer online.
- The product configuration is clear, and clearly differentiated from other random items on an order.
- If features and options have constraints (for example, if you select a particular transducer option, you must also select a compatible electronics layer), those constraints can be built into the configuration model.

Disadvantages:

- More mechanics in the system: you need a configurator and you need to define configuration models. The software is more complicated, and so are the recipes.

3. **Pre-configured product variations**: in other words, you offer a set of packages of configured options, or a set of different products for sale that include packages of configured options. You might have encountered these if you bought a car, where (for example) if you want cruise control, you must buy a higher-priced car model which also includes a bunch of other options like electronic locks and windows.

Advantages:

- No configurator or configuration models needed: the configurations are set in stone.
- The selling company may unload all kinds of undesired crap on the customer for profit.
- But pre-configured products might be good for customers if the configurations are motivated by the end use. For example, if a particular end use requires a particular configuration, then having it offered for sale as a package would simplify life for the customer, who would not need to know how to select the required options.

Disadvantages:

- Inflexible: the configurations are set in stone.
- Customers will hate them if they are motivated by profit-seeking rather than the end use.

4. **Combinations**: You could offer options a la carte as well as features with options as well as pre-configured product variations, as in a restaurant menu.

Sharing by design

[From **Yasir**]

Design products that are easy to share. We're building products for a sharing economy. What is a product that is easily shareable? What are its main features? What makes a product easily shareable?

- Modular because some users might be interested in some of its functions, not all.
- Standardization and interoperability because it can be used in conjunction with

other products.
- Customization - I can make it fit my needs.
- Durability and easy to maintain or repair.
- Trackability.
- Adaptive and reconfigurable, easy to calibrate, plug and play (even within a system of devices).

Culture and design

[From **Yasir**]

> In order to create products with certain features requires an established culture within the community of designers. It also requires practice, experience.

Warranty

See THIS great discussion about warranty. This is our first serious discussion on this topic.
Please curate it here.

Historical background

The idea to create this document came after a discussion between Tibi and Francois about planned obsolescence and the trend in software of ever-beta products.