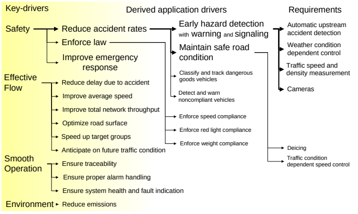
Key Drivers How To

-


Key-drivers
Derived application drivers
Requirements
Safety
Reduce accident rates
Enforce law
Improve emergency response
Effective Flow
Reduce delay due to accident
Improve average speed
Improve total network throughput
Optimize road surface
Speed up target groups
Anticipate on future traffic condition
Smooth Operation
Ensure traceability
Ensure proper alarm handling
Ensure system health and fault indication
Environment
Reduce emissions
Early hazard detection with warning and signaling
Maintain safe road condition
Classify and track dangerous goods vehicles
Detect and warn noncompliant vehicles
Enforce speed compliance
Enforce red light compliance
Enforce weight compliance
Automatic upstream accident detection
Weather condition dependent control
Traffic speed and density measurement
Cameras
Deicing
Traffic condition dependent speed control

Note: the graph is only partially elaborated
for application drivers and requirements

Gerrit Muller
University of South-Eastern Norway-NISE
Hasbergsvei 36 P.O. Box 235, NO-3603 Kongsberg Norway
gaudisite@gmail.com

This paper has been integrated in the book "Systems Architecting: A Business Perspective", `http://www.gaudisite.nl/SABP.html`, published by CRC Press in 2011.

Abstract

The notion of ”business key drivers” is introduced and a method is described to link these key drivers to the product specification.

version: 0.2 status: draft June 21, 2020

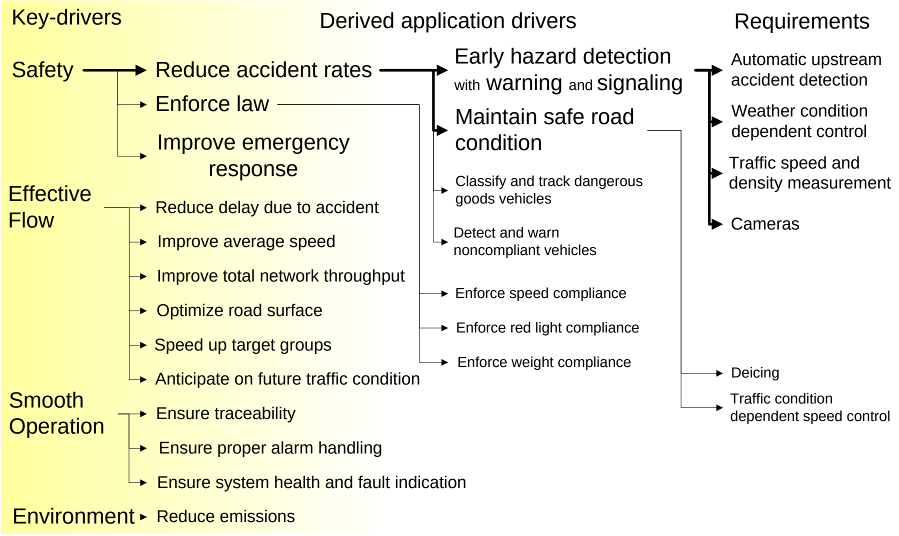
1 Introduction

A key driver graph is a graph that relates the key drivers (the essential needs) of the customer with the requirements in the product specification. This graph helps to understand the customer better, and the graph helps to assess the importance of requirements. The combination of customer understanding and value assessment makes the graph into an instrument to lead the project.

We will discuss one example, a Motor way management system, and we will discuss a method to create a customer key driver graph.

2 Example Motor Way Management

In this section we discuss an example from practice. The graph discussed here was created in 2000 by a group of marketing managers and systems architects. Creating this version took a few days. Note that we only show and discuss a small part of the entire graph to prevent overload.


Key-drivers
Derived application drivers
Requirements
Safety
Reduce accident rates
Enforce law
Improve emergency response
Effective Flow
Reduce delay due to accident
Improve average speed
Improve total network throughput
Optimize road surface
Speed up target groups
Anticipate on future traffic condition
Smooth Operation
Ensure traceability
Ensure proper alarm handling
Ensure system health and fault indication
Environment
Reduce emissions
Early hazard detection with warning and signaling
Maintain safe road condition
Classify and track dangerous goods vehicles
Detect and warn noncompliant vehicles
Enforce speed compliance
Enforce red light compliance
Enforce weight compliance
Automatic upstream accident detection
Weather condition dependent control
Traffic speed and density measurement
Cameras
Deicing
Traffic condition dependent speed control

Note: the graph is only partially elaborated for application drivers and requirements

Figure 1: The key driver graph of a Motor way Management System

Figure 1 shows an example of a key driver graph of a motor way management system. A motor way management system is a system that provides information to traffic controllers, and it allows traffic controllers to take measures on the road or to inform drivers on the road. As driver we typically see electronic information and traffic signs that are part of these systems. Also the cameras along the road are part of such system.

The key drivers of a motor way management owner are:

Safety for all people on the road: drivers and road maintainers.

Effective Flow of the traffic.

Smooth Operation of the motor way management.

Environment such as low emissions.

To realize these key drivers the owner applies a number of application processes. For example the traffic controllers can improve safety by reducing the accident rate. The accident rate can be reduced by detecting hazards and warning drivers about the hazards. Examples of hazards are accidents that already have happened and in turn may trigger new accidents. Another example of a hazard are bad weather conditions. Hence the automatic detection of accidents and controls that are weather dependent will help to cope with hazards, and hence will reduce accident rates and improve the safety.

Note that the 4 key drivers shown here are the key drivers of the motor way management system. Other systems will also share these concerns, but might not have these as key drivers. For example, smart phones will have a completely different set of key drivers. Do not use this example as template for your own key driver graph, because it biases the effort.

3 CAF-views and Key Drivers

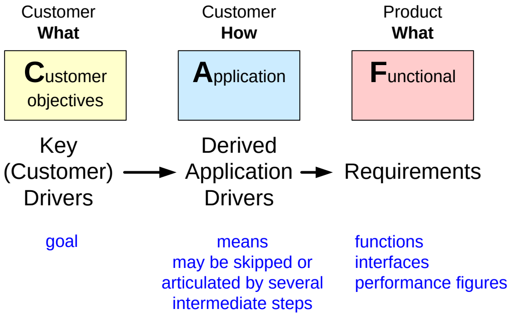
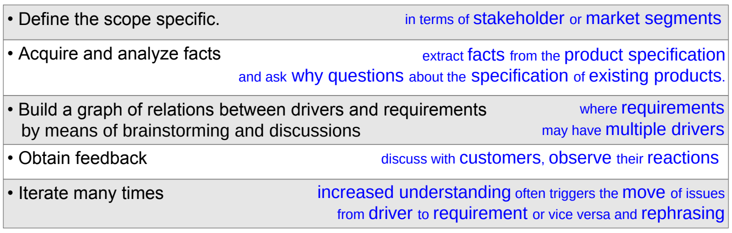


Figure 2: The flow from Key Drivers via derived application drivers to requirements

We can capture the essence of the customer world in the *Customer Objectives* view of the CAFCR model by means of customer key drivers. The customer will organize the way of working such that these key drivers are achieved. Figure 2 shows how the key drivers as part of the *Customer Objectives* view are supported by application drivers. The application drivers are means to satisfy the customer key drivers. These application drivers in turn will partially be fulfilled by the system-of-interest. Appropriate requirements, e.g. specific functions, interfaces or performance figures, of the system-of-interest will help the customer to use the system to satisfy their customer key drivers. The key drivers are one of the submethods in the Customer Objectives view.



• Define the scope specific.	in terms of stakeholder or market segments
• Acquire and analyze facts	extract facts from the product specification and ask why questions about the specification of existing products.
• Build a graph of relations between drivers and requirements by means of brainstorming and discussions	where requirements may have multiple drivers
• Obtain feedback	discuss with customers, observe their reactions
• Iterate many times	increased understanding often triggers the move of issues from driver to requirement or vice versa and rephrasing

Figure 3: Method to define key drivers

Figure 3 shows a method to define key drivers.

Define the scope specific . Identify a specific customer and within the customer a specific stakeholder to make the graph. Choosing a customer implies choosing a market segment. A narrow well defined scope results in a more clear understanding of the customer. The method can be repeated a few times to understand other customers/stakeholders. Products normally have to serve a class of customers. A common pitfall is that the project team too early "averages" the needs and by averaging compromises the value for specific customers. We recommend to first create some understanding of the target customers before any compromising takes place.

Acquire and analyze facts We recommend to start building the graph by looking for known facts. For example, in most organizations there is already an extensive draft product specification, with many proposed requirements. For every requirement in the draft specification the *why* question can be asked: "Why does the customer need this feature, what will the customer do with this feature?". Repeating the *why* question relates the requirement in a few steps to a (potential) key driver.

Note that starting with facts often means working bottom-up[1]. When marketing

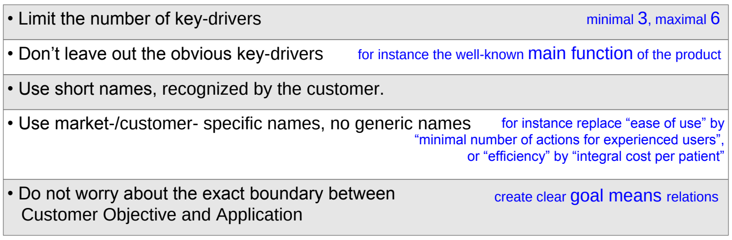
[1] Every time that course participants ignore this recommendation, and start top-down while lacking customer insight, they come up with a set of too abstract not usable key drivers.

and application managers have a good understanding of the customer, then the facts can also be found in the CA-views, allowing a more top-down approach. Iteration, repeated top-down and bottom-up discussions, is necessary in either case.

Build a graph of relations between drivers and requirements by means of brainstorms and discussions. A great deal of the value of this method is in this discussion, where team members create a shared understanding of the customer and the product specification. Note that the graph is often many-to-many: one requirement can serve multiple key drivers, and one key driver results in many different requirements.

Obtain feedback from customers by showing them the graph and by discussing the graph. Note that it is a good sign when customers dispute the graph, since the graph in that case apparently is understandable. When customers say that the graph is OK, then that is often a bad sign, mostly showing that the customer is polite.

Iterate many times top-down and bottom-up. During these iteration it is quite normal that issues move left to right or opposite due to increased understanding. It is also quite normal that issues are rephrased to sharpen and clarify.



• Limit the number of key-drivers	minimal 3, maximal 6
• Don't leave out the obvious key-drivers	for instance the well-known main function of the product
• Use short names, recognized by the customer.	
• Use market-/customer- specific names, no generic names	for instance replace "ease of use" by "minimal number of actions for experienced users", or "efficiency" by "integral cost per patient"
• Do not worry about the exact boundary between Customer Objective and Application	create clear goal means relations

Figure 4: Recommendations when defining key drivers

Figure 4 shows some recommendations with respect to the definition of key drivers.

Limit the number of key drivers to maximal 6 and minimal 3. A maximum of 6 Key Drivers is recommended to maintain focus on the essence, the name is on purpose **Key** driver. The minimum (three) avoids oversimplification, and it helps to identify the inherent tensions in the customer world. In real life we always have to balance objectives. For example, we have a strong need to maximize safety and performance, while at the same time we will have cost pressure. A good set of key drivers captures also the main tensions from customer perspective.

Do not leave out the obvious key drivers such as the main function of the product. For example, the communication must be recognizable when discussing smart phones; the focus might be on all kinds of innovative features and services, while the main function is forgotten.

Use short names, recognized by the customer. Key drivers must be expressed in the language of the customer so that customers recognize and understand them. The risk in teams of engineers is that the terminology drifts away and becomes too abstract or too analytical. Another risk is that descriptions or sentences are used in the graph to explain what is meant. These clarifying texts should not be in the graph itself, because the overview function of the graph gets lost. The challenge is to find short labels that resonate with customers.

Use market/customer specific names, no generic names . The more specific a name or label is, the more it helps in understanding. Generic names facilitate the "escape" of diving into the customer world. For example, the term *ease of use* is way too much of a motherhood statement. Instead *minimal number of actions (for experienced users)* might be the real issue.

Allocation to Customer Objectives or Application View Do not worry about the exact boundary between Customer Objective and Application The purpose of the graph is to get a clear separation of goals and means, where goals and means are recursive: an application driver is a means to achieve the customer key driver, and at the same time it is a goal for the functions of the system of interest. Sometimes we need five steps to relate customer key drivers to requirements, sometimes the relation is obvious and is directly linked. The CAFCR model is a means to think about the architecture, it is not a goal to fit everything right in the different views!

References

[1] Gerrit Muller. The system architecture homepage. `http://www.gaudisite.nl/index.html`, 1999.

History

Version: 0.2, date: December 16, 2010 changed by: Gerrit Muller

- corrected date of example (1998->2000)

Version: 0.1, date: June 29, 2010 changed by: Gerrit Muller

- removed CAFCR introduction from this paper
- added key driver introduction
- added text about method and recommendations
- textual updates
- changed status to draft

Version: 0, date: November 21, 2006 changed by: Gerrit Muller

- Created by taking a subset of “Requirements”