

# Managing Variability in ALPR Software

**Dr. Marco Sinnema**

Product Manager Video and ALPR, Q-Free ASA  
P.O. Box 180, 9410 AD Beilen, The Netherlands  
tel. +31 593 542055, fax. +31 593 542098  
marco.sinnema@q-free.com

## ABSTRACT

One of the challenges in maintaining ALPR software is to address the wide variation in customers' needs. Each specific ITS market segment implies different requirements on the recognition performance, for instance involving the read rate, error rate and memory consumption. In addition, variation in camera characteristics, weather conditions, and the variety in license plate layouts used around the world, imply even more variability to deal with in the OCR software components. In this paper, we present a successful approach to dealing with the wide range of required ALPR variability we experience in the ITS domain.

## INTRODUCTION

The term Automatic License Plate Recognition (ALPR) refers to the technology for automatically reading vehicle registration numbers from video or still images. This OCR technology is used in various ITS applications and markets, e.g. video tolling, toll collection enforcement, access control, red-light and speed enforcement, and measuring traffic statistics. ALPR is usually integrated into ITS systems, where the ALPR component reads road images or video frames and returns the car's registration number and country of origin. To save development cost of such ALPR components, systems delivered for the different ITS markets share as much technology as possible. Reusing (parts of) the OCR software, however, is difficult due to the implied variation in requirements on the functionality and quality of the ALPR products.

Since the 1990s, a successful approach to software reuse is software product families (1) (3), where the exploitation of similarities and supported differences over various products and markets is explicitly planned. The idea behind software product families is to break development down into two different processes, i.e. generic development and specific development. Generic development involves identifying commonalities and differences between product family members and realizing a set of shared software components in such a

way that the commonalities can be exploited economically, while at the same time the ability to vary the products is preserved. During specific development, individual products are derived from the product family using the software components realized as part of the generic development.

One of the suppliers of ALPR technology is the company Q-Free. In this paper, we explain how Q-Free adopted the software product family approach for developing their ALPR products for the various ITS markets. The figure below shows a screenshot of the Q-Free OCR technology as part of an ALPR application. The image in the top-left corner is processed in the application and the recognized registration number (69KXP4) and country of origin (NL) is shown in the center of the window.

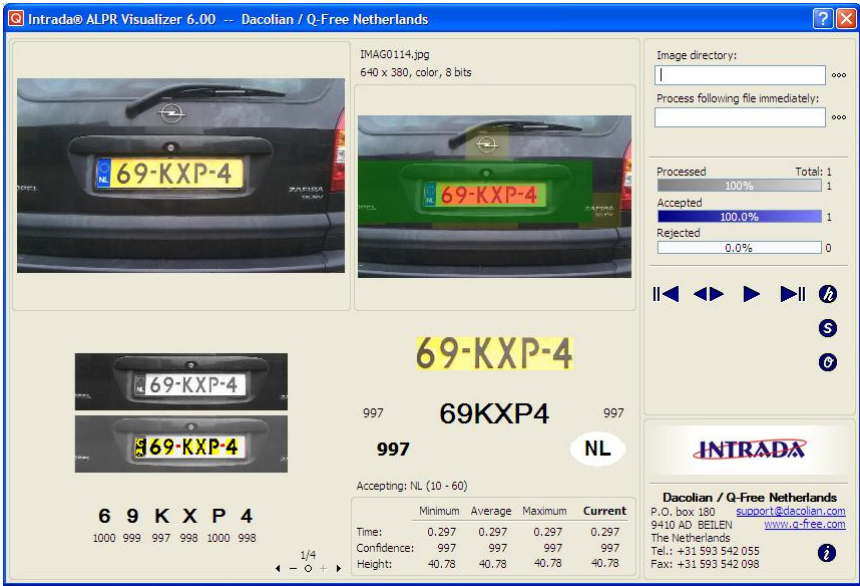
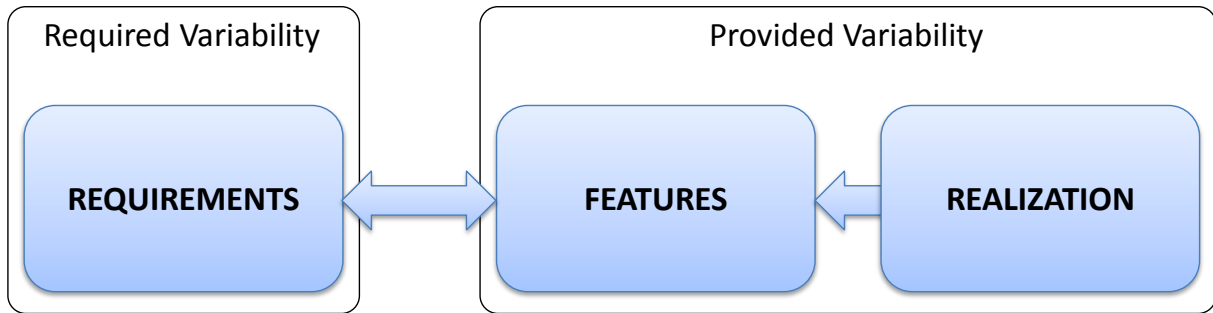


Figure 1 A screenshot of an ALPR application based on Q-Free OCR technology.

## VARIABILITY

As explained in the previous section, we have adopted the product family approach for developing and maintaining ALPR software. In addition to the exploitation of similarities between products, a key aspect of the software product family approach is dealing with the differences between products, also referred to as variability. Dealing with these differences requires a well-defined strategy for matching the variability *required* by the market, i.e. differences between the requirements from different customers and markets, and the variability in supported features *provided* by the software, i.e. the differences between the products allowed for in the realization. See also Figure 2.

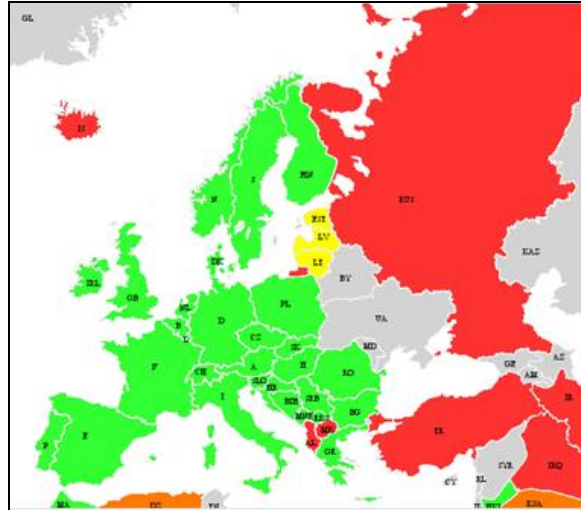


**Figure 2** As part of the product family strategy, the variation in Features supported by the Realization (provided variability), has to be matched with the variation in the Requirements from the different customers and markets (required variability).

For ALPR, variability from the customer’s point of view involves, e.g., the hardware/software platform, recognition performance, maximum error rate and supported countries, where the customer needs depend strongly on the respective market segment. Below, we briefly summarize the main variable aspects we experience in the customer requirements.

- **Performance:** With respect recognition performance, ALPR requirements mainly involve 1) the percentage of images correctly recognized, 2) the percentage of images incorrectly recognized, and 3) the percentage of images that are rejected. Whereas a low amount of rejected images is important in the parking and surveillance market, enforcement and tolling projects are more focused on experiencing a low number of incorrect recognitions.
- **Image quality:** Due to weather conditions and camera characteristics, input video shows a wide variety in resolution, compression, and contrast to be supported.
- **Countries:** The countries around the world all use a different layout of the registration plates. For ALPR, customer needs usually contain requirements on one or more countries to be supported.
- **Technical:** In addition to the pure ALPR aspects mentioned in the previous bullets, also variation in hardware platform, processing power, operating system and available memory needs to be address within the software product.

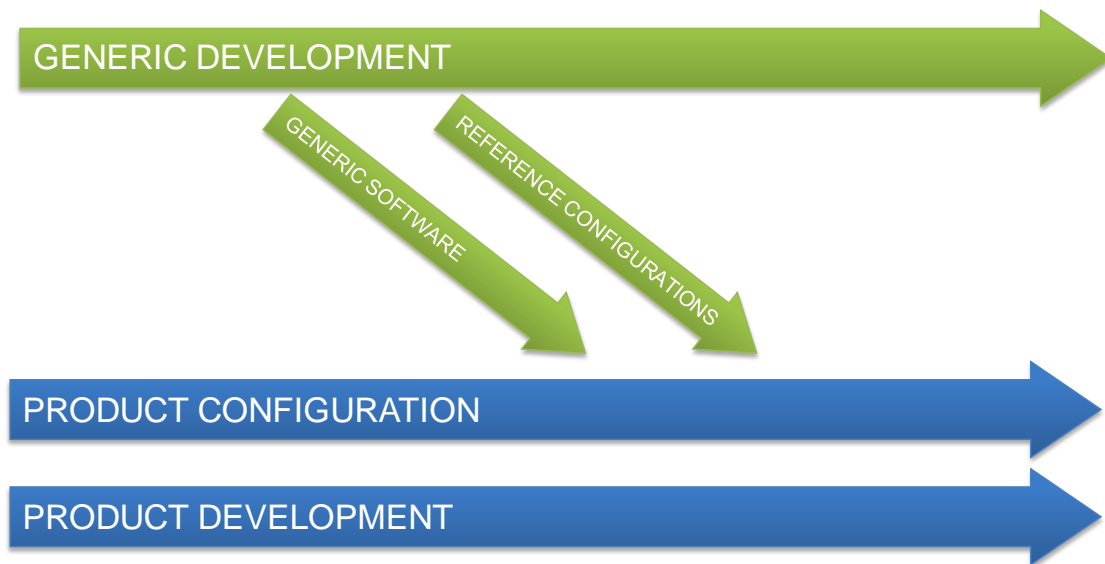
To be able to address the variable needs from the market, the aforementioned required variability should also be provided by the software product family artifacts. This holds for the recognition performance, the image quality, the supported countries, as well as any technical aspects. As an example, we show the variant features related to the third bullet (countries) from the ALPR product family being developed by Q-Free in Figure 3. To address the variation in countries and plates to be read in a product, the selection of supported countries is one of the features provided by the product family. For the realization, this means that various (combinations of) plate layouts need to be supported.



**Figure 3 Visualization of available variants, the supported countries.**

## DEVELOPMENT PROCESSES

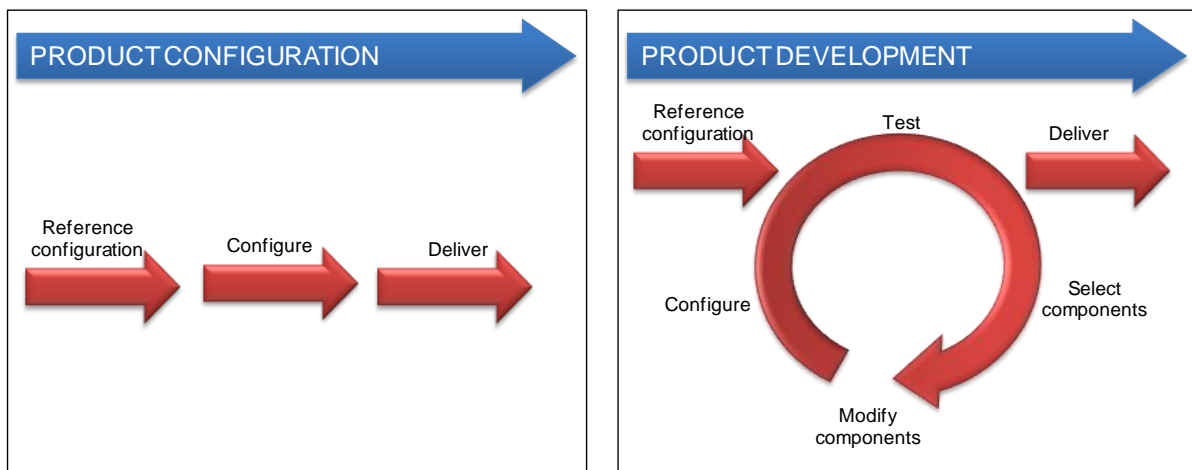
Following the established research on product family engineering (2), we also distinguish between generic and specific software development, to implement the strategy explained in the previous section. These two processes are visualized in Figure 4, where generic development is marked by green arrows and specific development is marked by blue arrows.



**Figure 4 Product family engineering at Q-Free breaks down into formally implementing generic and specific development, where we distinguish between product configuration and product development.**

To be able to ALPR software as a product, the software modules need to be configurable in such a way that they provide the features (and thereby the requirements) mentioned in the previous section. One of the challenges to deal with is that these aspects cannot be configured

independently. E.g., a hardware platform within the physical camera implies restricted processing power, thereby limiting the recognition performance that can be achieved. Even more challenging is the dependency between the three aspects of recognition performance, where configuring for a lower number of rejected images will usually have a negative impact on the quality of the recognition (correct/incorrect). In general, optimizing a product with respect to one aspect will very likely have some negative impact on another. As a result, tuning a high performance product is a time-consuming task, not possible to address the needs for all different market segments.



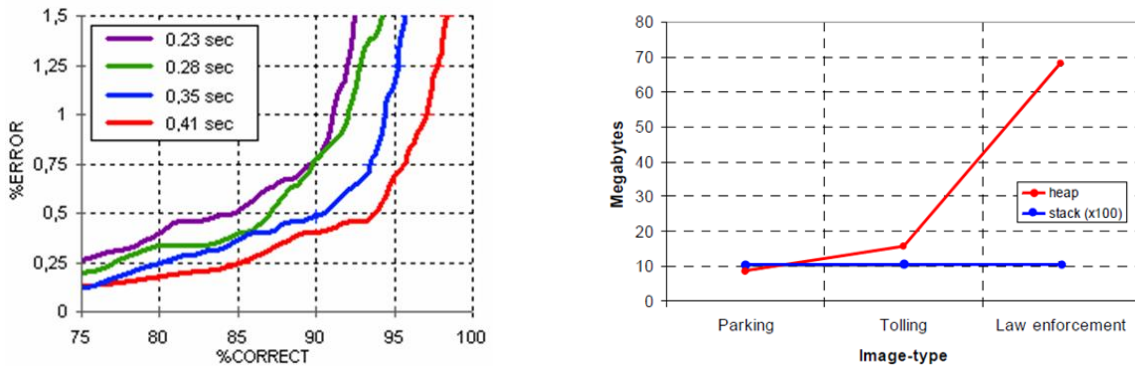
**Figure 5 To be able to deliver standard products, while also allowing to develop tuned systems, we differentiate between Product Configuration and Product Development processes.**

In order to distribute the effort over these market segments anyway, we explicitly distinguish between product configuration and product development, shown in Figure 4 and Figure 5.

- Product Configuration:** The generic software provided by generic development is used as-is to develop standard ALPR products, for example to be used in a system for measuring traffic statistics. Reference configurations on this generic software show the various performance characteristics, given a set of configuration parameters. Developing these reference configurations is part of generic development. Deriving a product via product configuration means we apply a straightforward process to go from a reference configuration to a delivery, only consisting of selecting the hardware platform, operating system and countries to be supported. This process is visualized in the left part of Figure 5. This straightforward process evidently implies a restriction on the possible performance and image quality requirements we may be able to address with such products, and thereby limits the possible target market segments.
- Product Development:** To develop a high performance ALPR product, e.g. for video tolling, we start by taking an appropriate reference configuration. In multiple iterations of development, testing, and configuration, we subsequently get to a product that fully

satisfies all customer requirements on the recognition performance and image quality as well. This iterative process is visualized in the right part of Figure 5. To deal with the aforementioned dependencies between various aspects, measurements from combinations of reference configurations are used to guide the development and configuration process. This allows to define a strategy that eventually leads to a suitable product. Preparing measurements on reference configurations, and thereby characterize the dependencies between configuration parameters, is part of the tasks performed in generic development (See also Figure 6). We refer to (4) for more detailed information on product development and dealing with dependencies in software product families.

Note that, although these two processes for specific development are very different, they both adhere to the generic two-phased product derivation process introduced in (3).



**Figure 6 Investigation of dependencies is necessary to decide on a strategy to derive a product that satisfies all requirements. On the left, the effect of varying maximum processing time on some ALPR configuration is shown. On the right, estimations of memory usage are shown in relations to different market segments.**

## CONCLUSION

To be able to address OCR requirements from various ITS applications and markets, Q-Free adopted the software product family approach to develop their automatic license plate recognition (ALPR) technology. As a result, development effort is being shared over different products, cost have been saved, and the quality of the software has increased. In this paper we explained how we have implemented this approach successfully. This implementation breaks down into matching the different customer requirements to the various features provided by the software components, a clear separation between generic and specific development, as well as how we explicitly distinguish between product configuration and product development.

## REFERENCES

- (1) Bosch, J, Design and Use of Software Architectures: Adopting and Evolving a Product Line Approach, Pearson Education (Addison-Wesley & ACM Press), ISBN 0-201-67494-7, 2000
- (2) Clements, P, Northrop, L, Software Product Lines: Practices and Patterns, SEI Series in Software Engineering, Addison-Wesley, ISBN: 0-201-70332-7, 2001
- (3) Deelstra, S, Sinnema, M, Bosch, J, Product Derivation in Software Product Families; A Case Study, *Journal of Systems and Software*, Volume 74, Issue 2, pp. 173-194, 2005.
- (4) Sinnema, M, Deelstra, S, Nijhuis, J, Bosch, J, Modeling Dependencies in Product Families with COVAMOF, Proceedings of the 13th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2006), pp. 299-307, 2006