Polar Case Study

"A detailed evaluation proved the benefits of the DSM solution: an **increase of at least 750%** in developer productivity, and **greatly improved quality** in the code and development process," Juha Kärnä from Polar.

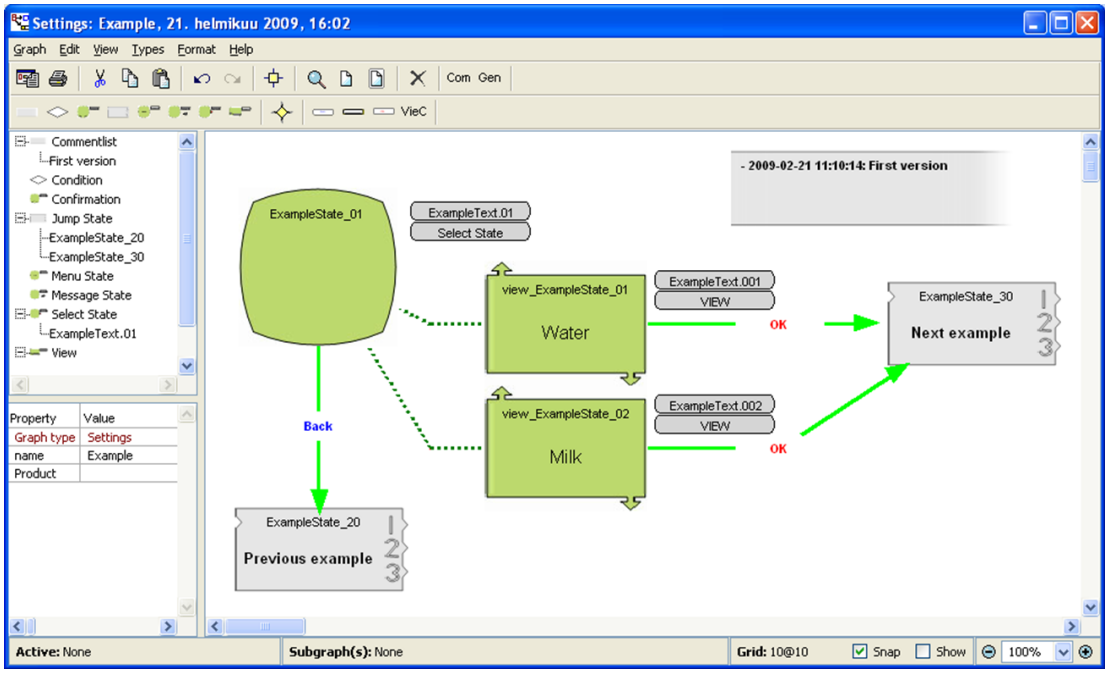
Polar is the leading brand in the sports instruments and heart rate monitoring category delivering state-of-the-art training technology and solutions. The features in a Polar product depend on the product segment and the type of sports the product is designed for, such as running, cycling, fitness and cross-training, team sports or snow sports.

Software development for these devices is constrained by the limited resources they contain, such as the amount of memory, processor speed and battery life. The actual area of interest — the domain — covered by the modeling and code generation is the UI applications: how the various capabilities and features are made available to the user. The design and implementation of the UI applications is heavily constrained by device capabilities such as display size, type, and user interaction controls. It is worth mentioning that as these devices are used in special conditions — users may have little time and concentration capability while exercising — the usability of UI applications is crucial.

Polar was looking for ways to fundamentally improve the productivity of UI application development as well as the quality and maintainability of the code. Other important requirements were usability, easy introduction and becoming independent of the target programming languages and environments.

THE SOLUTION

At Polar, one UI application developer defined the modeling language, along with the generators that transformed models made with that language into the artifacts the company needed (e.g. code, configuration files, links to simulators, document generation). The modeling language was supported by a tool, MetaEdit+, that provided the functionality needed to work effectively with models, such as reusing models, refactoring and replacing model elements, organizing and handling large models, multi-user access - as well as usual modeling operations like copy and paste.



UI application developers can thus use the modeling language and tool to create high-level models (see screenshot). The modeling language raises the level of abstraction from coding, while also providing support for reuse when developing multiple products. The diagram is also executable, in that full code can be automatically generated from it.

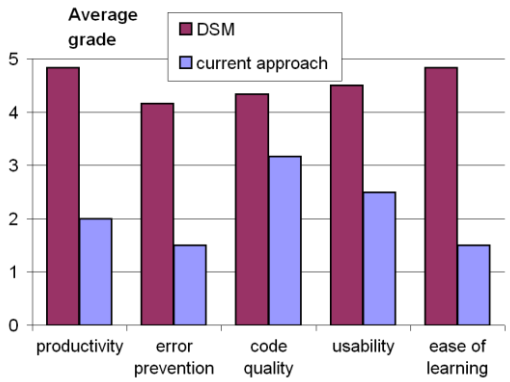
While the screenshot illustrates the use of the language, it is about the smallest possible model. In real cases there may be dozens of elements in a diagram, dozens of diagrams in an application, and dozens of applications in a full product. An element in one diagram can be linked, referred to and reused in other diagrams, or can be linked to a subdiagram specifying it in more detail. Applications too can be reused between products.

EVALUATION

The influence on productivity was inspected in two ways: by measuring the development time and by collecting developers' opinions with a questionnaire after having used both approaches: the current development method and the DSM approach used for the first time.

- The measurement revealed that the DSM solution was found to be at least 7.5 times and on average 10 times as productive as the current development approach.

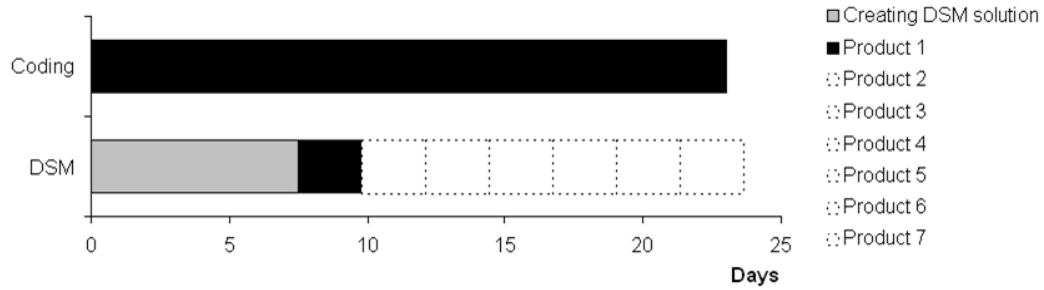
- In the questionnaire, the DSM solution was considered to offer better productivity, quality and usability, and be easier to learn. The histogram summarizes the questionnaire findings by comparing the current approach and DSM based on the average grading calculated from developers' opinions (scale 1-5, 5 best).



RETURN OF INVESTMENT

At Polar, creation of the DSM solution took 7.5 working days, covering the development of the modeling language and the code generator. Both of these were implemented using MetaEdit+ Workbench. MetaEdit+ automatically provides modeling tools based on the modeling language, so no extra time needed to be spent on tool building. It is worth noting that the 7.5 days also included the creation of example models specifying UI applications, along with related code. This was natural since the best way to test a DSM solution under development is to apply it immediately on real examples.

When we compare the time to implement the DSM solution to the productivity improvements when creating UI applications, it is evident that the investment pays back very quickly. With DSM, after the 7.5 days' metamodeling, the first whole product would take 2.3 days to build, making DSM over twice as fast as coding even for the first product. Each subsequent product would take another 2.3 days, so in the time it took to build one whole product by coding, Polar could build several whole products with DSM.



YOUR NEXT STEP

Visit us at http://www.metacase.com to see how MetaEdit+ can speed up your software development!

Source: Kärnä, J., et al. Evaluating the Use of Domain-Specific Modeling in Practice. OOPSLA Workshop on Domain-Specific Modeling, 2009, http://www.dsmforum.org/events/DSM09/Papers/Karna.pdf

MetaCase
info@metacase.com
www.metacase.com