Reuse in the car industry

The car industry has many many differences to the software industry. Most obviously production costs; once your magnum opus software is complete production costs are minimal - CD duplication or server bandwidth. In the motor industry completing the design is but the first stage: production costs dominate. In recent years *just in time* assembly lines and *lean* practices have revolutionised car production, while increased competition between manufactures has lead to increased product diversification and a reduction in the lifetime of models. Consequently, competitive advantage rests with those manufactures that can produce a diverse range of frequently updated models. This places the onus back on design.

In an effort to reduce design costs manufactures have turned to reuse: inside seven years Volkswagen moved from making 30 different models on 16 floorpans to 54 models on 4 floorpans[1], look how the Passat is an Audi A4, is a Skoda Octavia.

With this background Cusumano and Nobeoka[2] have looked at the design process, and in particular how design reuse is enabling car companies to meet this challenge. Although *Thinking beyond lean* was published in 1998 its lessons are not dated and provide valuable insights into the reusing design and transferring technology between projects. Almost as vindication, since 1998 we have seen platform sharing as a major factor behind the Renault-Nissan, Mercedes-Chrysler-Mitsubishi and Ford-Mazda-Volvo link ups.

Cusumano and Nobeoka identify four development models, three of involve transferring technology:

? New design : "projects that primarily develop platforms from scratch"

? Concurrent technology transfer: "a new project begins to borrow a platform from a base preceding project before the base project has completed the design work. Generally, this transfer occurs within two years"

? Sequential technology transfer : "a project inherits a platform from a base project that has finished the design work"

? Design modification : "a project that replaces an existing product without creating a new platform or borrowing a platform from another production line"

I suggest that these models hold true for software development. A car *platform* differs from a software *framework* in that a platform is built for a specific product but may be reused with modification; while a software framework is designed from the start as a generic environment. Platforms provide greater focus on the initial project.

Cusumano and Nobeoka concentrate on Toyota, and their use of development centres and concurrent technology transfer of platforms between Toyota and Lexus lines. Honda uses more traditional matrix management practices but has great success sharing common platforms between more diverse products – witness the Civic platform, which is also used for the CRX sports car and CRV SUV while internationally the same platform was used by Rover for several years. Elsewhere Chrysler's continued use of sequential transfer and design modification in the 1990's reduced short-run costs but lead to an outdated product line with problems competing in the market.

From analysis to coding, software development is an exercise in design so parallels with car design are highly relevant. Perhaps the biggest lesson is: reuse does not just happen, you have to manage it, you have to put processes in place to deliver reuse. (Which brings us back to Conway's law – "align process and design.")

My experience is that most code-level reuse is bottom-up, it comes from engineers designing a part to be reusable. Cusumano and Nobeoka show how management can be aligned to this process and how they can encourage it. If you want reusable software you must align your process, engineers know how to reuse, management must be aligned.

Listings the lessons of Cusumano and Nobeoka would take an article in their own right, better you go and read the book, however, here are a few ideas:

? Look to design a product with a platform; plan for the platform to be re-used, follow-on projects should overlap with the original project

? Organise around product lines not functional abilities

? Use strong, autonomous, project managers to drive projects forward

? Physically co-locate project teams together

? Integrate R&D with product development

? Keep communication complexity to a manageable level

? The "company memory" approach does not work well, leading to reuse of old components and infrequent replacement cycles

? Informal "tech clubs" can be used to promote reuse without formal multi-project management.

? Written documentation is good for transferring component knowledge but poor at transferring integrative knowledge; perhaps worryingly, evidence shows that placing a lot of emphasis on retaining prior knowledge can hamper innovation and use of new technologies

Finally, sometimes we need to forget about reuse, sometimes reinventing the wheel brings benefits. Mazda's Miata (MX-5) was developed by a "guerrilla team", physically separated from the main development groups, freed from reuse restraints, with a mandate to produce a car to enhance Mazda's image. Reuse is a valuable tool, but it is not an end in its own right.

Few, if any, software projects are on the scale of designing a car, still, there are useful lessons and ideas here. The software industry is not an island, we must look to established business, management and engineering models to understand and improve ourselves.

[1] *The Economist*, 27 April 2000

[2] Cusumano & Nobeoka, *Thinking Beyond Lean*, Freepress, 1998