Two More Business Patterns:

CUSTOMER UNDERSTANDING and CUSTOMISABLE PRODUCT (EuroPLoP 2011)

Allan Kelly, allan@allankelly.net

Abstract

This paper contains two additional business patterns to complement the authors existing body of work. These patterns are:

- CUSTOMER UNDERSTANDING: Go and meet your customers - both those who pay and those who use your products. Interview them, observe them, question them, listen to them. Look at who they are and what their relationship is with your product; how they use the product and why they use it.
- CUSTOMISABLE PRODUCTS: Allow customer specific modifications and changes to exist separately from the generic product base. Make it possible to customise the product without changing the generic source code.

1 Introduction

During the last seven years the author has presented multiple business patterns at EuroPLoP and VikingPLoP conferences (Kelly, 2004a, Kelly, 2004b, Kelly, 2005b, Kelly, 2005a, Kelly, 2006b, Kelly, 2006a, Kelly, 2007a, Kelly, 2007b, Kelly, 2008, Kelly, 2009). During 2010 these patterns were assembled into a single document, examined as a whole and in some cases reworked with a view to publishing the patterns as a book (Kelly, 2012).

Upon examining the whole several missing pieces were found. The most pressing of these are presented here as two patterns:

- CUSTOMER UNDERSTANDING
- CUSTOMISABLE PRODUCTS

These patterns are related in that they form part of a much larger pattern language of business strategy for software companies. However, within this language the two patterns are not closely related. They are presented here, together, for convenience.

In order to give each pattern more context the relevant pattern sequence diagrams are also presented - thumbnails for the patterns referenced in the sequences are to be found at the end.

2 Patterns about marketing

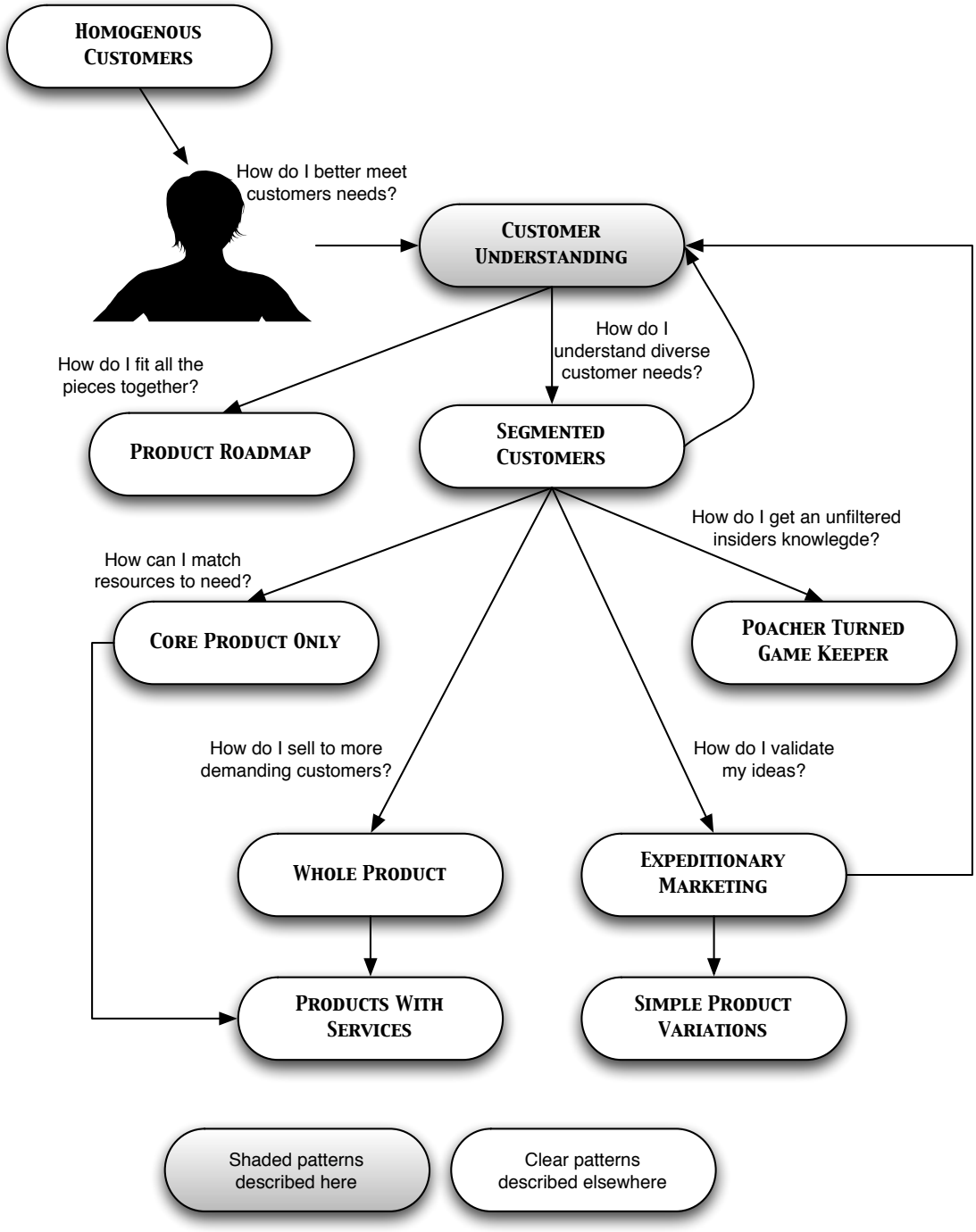

Figure 1 - Pattern sequence for marketing patterns (For pattern thumbnails see section 4)

2.1 CUSTOMER UNDERSTANDING

Figure 2 - Customer Understanding

Intuit Product Managers are famous for putting customer understanding at the heart of their work. They are reported to have watched customers select and buy Quicken in computer stores. They then approach the customer, ask about their reasons for choosing Quicken and on occasions accompany customers home and watch them install the product on their PC and observe how they used it.

Context You are developing a product and need to know what qualities and features the product needs - and what it doesn't need. Maybe you are not using CUSTOMER CO-CREATED PRODUCT or maybe you want to look beyond the lead customer.

Problem **How do you understand what your customers and the wider market want (and will pay for)?**

Forces Everyone has opinions and views on what a product should do, but desk bound engineers and managers are not representative of real customers. This is especially true in start-ups were past success may be based on founders intuition rather than research and inexperienced staff may be afraid to meet real customers.

Understanding customers in depth is difficult when they don't understand what the technology can do for them. Your engineers understand the technology but customers don't. *If you build it, they will come...* maybe, or you will be left with a costly product nobody wants to buy - like the HP TouchPad tablet. Technology is only useful if it can be applied to solve a problem or make life better.

You want to produce products customers want to buy but finding potential customers, meeting them and finding their problems takes time and money. It is easier to believe that if a company employs smart people, they will work out what customers want by themselves using logic and reason.

Customers are not homogenous in their needs or wants, and some customers have more buying power than others. But you want to create a product which will appear multiple customers - a market, not an individual.

Every sales person will tell you they know what the customer wants; sales people are often reluctant to let someone else deal with the customer. Sales people worry that non-sales folk may jeopardise a sale or relationship; and they worry that someone else might steal their customer (and their commission.) But to create a product that meets the needs of the wider market the views of individual customers must be balanced.

Solution

Go meet and observe your customers - those who pay, those who use your products and those you would like to have. Interview them, observe them, question them, listen to them. Look at who they are and what their relationship is with your product. Seek to understand both your customer and their environment.

Understand customers as people; *what is the average age of your customers? Their education level? How long do they (would they?) spend with your product in an average day? What does their day look like?*

Observe how customer actually use the product, rather than how you expect them to use it. *Are there parts they never use? Do they use it for things you don't expect? Do they understand why they like (or dislike) the product? Can you apply Occam's Razor or the Pareto Principle?*

These may be obvious questions when selling consumer products but they apply too when selling to corporations: organizations don't choose and use products, individuals do. Don't stop at company or group names, find those people who choose, use and benefit from your product in the organization.

When selling to organizations understand the culture and environment of the company. Differentiate between economic buyers (those who choose your product and agree to purchase based on financial grounds), technical buyers (who evaluate it and choose it for technical abilities) and actual users who may not have a direct voice in the purchase decision but use the product daily.

Meet with potential customers who evaluated your product but chose not to buy. Understand why they did not buy and what you might do in future to win them as customers. Repeat the exercise with those who did buy and undertake "Win/Loss analysis."

Find out what they think about your product, competitors, what their work and life entails and how it could be better: *why do they buy your product? Why do they use it? What problem does it solves for them? What constraints do they operate within?*

For business products there may well be a business driver (cost savings, greater efficiency, time to market) for using your product. For consumer products the benefits might be less tangible or easy to define - enjoyment, gossip, fashion.

Compare customer wants with their ability, and willingness to pay. Many desirable features may simply be uneconomic to develop. Conversely delivering specialist capabilities at a new price point may provide an advantage over competitors.

Understanding is not a single event. It requires constantly attention, there are always more customers, and potential customers, to talk to, and those you have talked to don't stand still. Visit them again; find out how their lives change and how your product changes it. Employ dedicated specialists to keep them under review.

Use qualitative techniques to understand customers and their motivations. When the customer base is large you might follow up with quantitative surveys. However, quantitative studies alone cannot lead to deep understanding of customers and their issues because aggregate answers cannot show intentions and motivations.

Use SEGMENTED CUSTOMERS to divide your customers into groups and consider each group individually. Examining each customer segment separately will deepen understanding which will in turn informs segmentation (Figure 3). When dealing with large organizational customer consider the range of stakeholders within the organization.

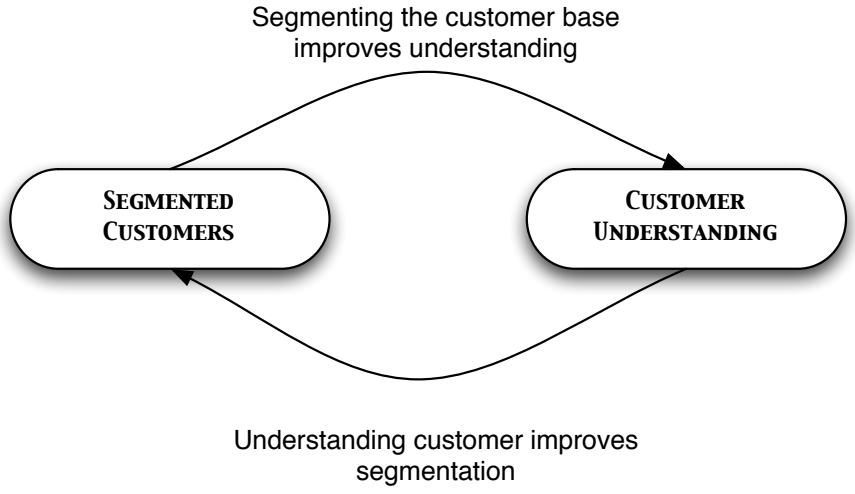

Figure 3 - Segmentation improves understanding which improves segmentation

When visiting customers who did not buy tell them "I'm not here to change your decision, but I'd like to know why you chose the

competitor so we can create a better product you might consider next time."

It can be difficult and time consuming to meet with customers so stage events where customers come and meet with you. This might be a "meet the team" event inside your company, or a user conference with many different customers. You might even invite one of your customers to come and brief the team on the work they do and how they use your product.

Stay in contact with customers once you have met them, see how their needs change and float ideas for product changes with them (though be careful not to commit yourself.) Once you have met the a customer the first time it is easier to talk openly on the telephone. You can stay in touch without the time and expense of a visit by using the phone.

Spend time thinking through what you want to know about customers and what you need to know to direct your product. Keep a list of standard questions which you can use to get beyond a customer immediate issues and stock responses. After you meet a customer review your meeting, document your findings and refine your questions for next time.

As your understanding of customers grows create personas (Cooper, 2004) to help communicate your findings and share your understanding with the rest of your company.

Consequences

In depth understanding of customers, their attributes, needs and wants provide a solid base for reasoning about product development. Facts about what customers need, how they use products and what they are trying to achieve (should) trump views and opinions every time.

When you understand what customers need you can apply your understanding of the technology available to create an innovative product. This will guide your product development and improve the chances of producing a star product.

Consequently your product will meet customers' needs better than competitor products which are based on technology alone, or the ability of a salesman to make a round peg fit in a square hole.

By segmenting the customer base Product Managers decide which customers wants and needs are considered first and foremost. Focusing on a limited number of customers helps create the right product but means some customers' needs are left to one side. Trying to please all customers equally not only makes product development harder but may result in a product which is ideal for no-one.

Sales people may object to others visiting "their customers" because they seek to control the customer relationship and don't want customers poached (even internally). They may object because "I already know what the customer wants". However, what one

customer wants might not be the thing the wider market wants and this may create tension.

Product Managers need to win the trust of Sales by demonstrating they do not want to steal the customer. Product Managers tasked with understanding the customers are not paid on commission and look beyond the next sale. They aim to create great products and leave selling to professional sales people.

It takes time, and money, to meet customers and understand the problems, constraints and opportunities they face. You also need the right skills to ask the questions and listen without putting words into the customer's mouths. In a technology based company it can be difficult to see the value of people who are not actually coding.

Variations

POACHER TURNED GAME KEEPER provides another solution which can lead to a focus on a particular segment. SURROGATE CUSTOMER (Coplien and Harrison, 2004) offers other solutions to the same problem.

Many of the techniques noted above can be used to analyse competitors products and their customers as well as your own.

Related work

KNOW THE CUSTOMER (Rising, 2002) pattern contains similar advice. Other patterns in Rising's series can also be used to help - IT'S A RELATIONSHIP, NOT A SALE, BUILD TRUST, MIND YOUR MANNERS and LISTEN, LISTEN, LISTEN

This pattern is the basis of the Product Manager role in independent software vendors and feeds the creation of the PRODUCT ROADMAP.

If you are using BRANDED SHOPS (or have good relations with an INDEPENDENT RETAILER) your product development staff may visit the stores, observe the sales process or even work in the store for a few days to improve their customer understanding.

SALES/TECHNICAL DOUBLE ACT can support this pattern by giving different view points on customers and the individuals inside a company.

Many organizations start with HOMOGENOUS CUSTOMERS implicitly and after understanding their customers arrive at SEGMENTED CUSTOMERS.

Techniques for increasing customer understanding are also discussed in *Tuned In* (Stull et al., 2008), *Inspired* (Cagan, 2008) and on product management training courses.

Examples

Visiting customers to understand their relationship and need for your product is an example of the Lean idea of *Gemba* - "the place of truth," "where the work happens." (Womack and Jones, 2005)

3 Patterns about Products

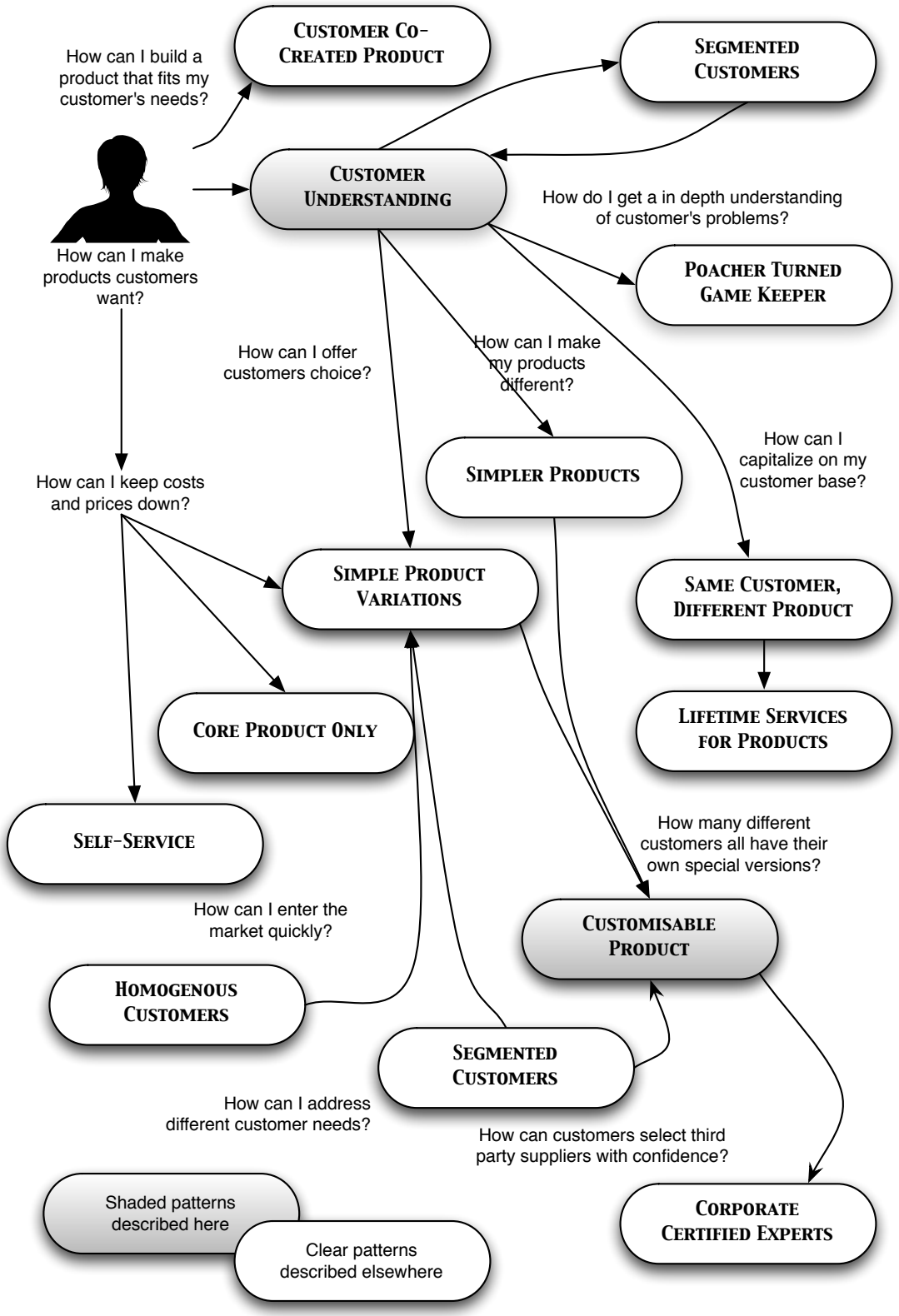

Figure 4 - Product patterns map (For pattern thumbnails see section 4)

3.1 CUSTOMISABLE PRODUCT

Figure 5 - Cars get modified in all sorts of ways

Some companies set out to build platform products and some do it, almost, by accident. Nokia's Symbian operating system was always designed as a platform and initially attracted dedicated developers. Apple's iPhone was (almost) an accidental platform, according to some reports Apple management resisted idea for the AppStore and did not see the potential.

Context You are an ISV[1] with an existing software product in the market (or near release). A customer has approached you with a special request.

Problem **How do you incorporate customer specific functionality without creating multiple products variations?**

Forces You have a generic product available to multiple customers but some (potential) customers would like a special version with features and functionality just for themselves.

The functionality requested is more than a few options in the program preferences and the changes requested go beyond SIMPLE PRODUCT VARIATIONS but are of no interest to other customers or might even detract from their ownership experience.

The customer wants to control the intellectual property (IP) in the "special" version - they may wish to include their own IP in this

[1] Independent Software Vendor

version, or to take ownership of the IP created by your team, or simply to restrict competitors access to the IP and changes. Consequently you cannot sell the changes to other customers and CUSTOMER CO-CREATED PRODUCT is not an option.

You would like to carry out more customer customisation work. But, when such work involves modifications to the core product source code. As the number of specials increases there is a combinatory explosion: *Which modifications work with which others? Which combinations should be tested? How are priorities to be set? And how should you control different the source code bases?*

There are many ways engineering can produce multiple, similar but different version of a product: they could branch the product, use compiler macros, or configuration options. But taking this route rapidly leads to a multiplicity of options, complicating development, testing and increasing the maintenance workload.

Incorporating these changes into your existing product will mean diverting your PRODUCT ROADMAP to satisfy the needs of one customer, thus penalising customers of the generic product. But, the customer in question is willing to pay good money for the special changes. For a start-up company the money on offer might be too good to turn down; for an established company the money will add significantly to profit.

Solution

Allow customer specific modifications and changes to exist separately from the generic product base. Make it possible to customise the product without changing the generic source code.

Set clear dividing lines between what is generic product work and what is customer specific work. These lines need to exist on the PRODUCT ROADMAP (which is being diverted), in the organisation and management of the work and in the code base.

Initially it is best to have your own development team carry out customisation for customers using the new API, scripts, etc. They are the people who know most about how the product can be customised and doing customisation will improve their understanding of how well the features work. They have a vested interest in keeping generic and customer specific pieces cleanly separated, but on the flip side they know how hack it together too.

If customisation projects are happening regularly it is best to establish a dedicated team for this work and introduce more specific variation points into the code base (see sidebar *Providing customization points*). This will allow core developers to continue working on the generic product without the disruption of special projects. More importantly separate teams will create a division between special work and generic work. This will prevent special functionality appearing in the generic product and safeguard IP. Conway's Law (Coplien and Harrison, 2004) predicts that separate

teams will create separate designs.

Over time some customer specific feature may migrate to the common product and some common features to customer specific enhancements. Take care not to infringe IP when making customer specific items common to all.

Consequences Separating the generic product from the special customisations allows the generic product to follow its own roadmap while the customised version meet specific customers' needs.

Creating a barrier to separate the generic product and the customer specific customisations allows the two sides to develop and change at difference speeds.

Code and customisations on the far customer side of the barrier can be controlled and stored separately from the generic product. Since customisation work is isolated it does not cause a combinatory explosion of products or product configuration.

Customers only see their customised version. Neither they nor the core development team are encumbered with multiple special cases.

Providing customisation hooks is a major undertaking and will need to be included on the PRODUCT ROADMAP before the work is done. Once the hooks are provided customisation work will not need to be included on the product roadmap and should not cause further diversions.

Variations CUSTOMISABLE PRODUCT IS the first step towards creating a platform rather than a product. Platforms, like operating systems exist to be customised for other uses. The line between a customisable product and a platform is particularly blurred when considering devices like the Apple iPhone and Microsoft XBox. Customers can download additional applications from an *App Store*. This trend will become more prevalent as digital TVs become platforms and gain their own app stores.

Related work When customisation work becomes too much for your own resources to handle, or you want to encourage third parties to get involved use CERTIFIED CORPORATE EXPERTS. Doing this will help create an ecosystem to develop around your product.

This patterns differs from SIMPLE PRODUCT VARIATIONS and *Mass customization* (Pine, 1993) because it allows for far more, potentially unlimited, customization.

When customers are willing to share enhancements and IP CUSTOMER CO-CREATED PRODUCT is an alternative pattern with lower costs. Providing customisation services to customers is a form of PRODUCTS WITH SERVICES.

Customer requests for special versions is a reoccurring theme in the ISV world and posses particular dilemma's for early stage cash-starved companies.

Cagan strongly advises against agreeing to special versions, of

products which are not intended to be platforms (Cagan, 2008).

Examples Many well-known software products provide for customisation. Microsoft Excel provides a scripting language, a COM API allowing the application to be driven from another program, and a plug in API allowing users to install new functions. The Emacs editor goes even further by allowing users to add and even change the Lisp implementation of the program is written in.

Customisation, Platforms, Frameworks and Product Lines

Software is infinitely flexible but it doesn't always make sense to take flexibility to the extreme - both for technical and marketing reasons.

The pattern CUSTOMISABLE PRODUCT is about making modifications to a standard application product so it better meets specific customer's needs. As described it is a solution to a specific problem with specific forces.

There are good reasons to argue against this approach, this author, in general, agrees with Cagan's argument against "customer specials" (Cagan, 2008). However, "specials" are common practice and many companies do them; simply saying "don't do it" is not useful advice for many.

There are other forces which can lead to similar solutions. For example, WHOLE PRODUCT advocates making the product fit not just a single customers need exactly but making product that perfect fits the needs of an entire industry, or market.

Economics exerts powerful forces too: *Why design, manufacture and maintain a product for a small niche market when by targeting a larger market those costs can be spread over many more customers and sales?*

But in addressing bigger and bigger markets the number of points of variation increases. This gives rise to more complexity, both in the underlying code, in the customisations and the marketing message to potential customers.

Attempting to manage these complexities has given rise to Product Line Engineering and the idea of technology platforms on which applications can be built. Other commercial/engineering domains have pursed platform approaches and there are lessons to be learned.

For example, car manufactures build product lines for different car models: there is not just one type of Volkswagen Golf but several, the popular hatch-back (with two or four doors), an estate version and a saloon version (sometimes called Bora or Jetta.).

But car manufactured don't stop there. Increasingly more than one product line of car is built on the same platforms. Underlying the VW Golf is the PQ35 platform which also serves for the Audi A3, Skoda Octavia and other cars in the VW stable - each of which has its own product line of saloons, hatch-backs, and so on.

Owners may then choose to individually customises their cars - a special paint job, or a fancy spoiler; or structural changes, wheel chair access or foot steering for a disabled driver.

Some software companies have been successful in adopting this model but history is littered with failed examples. The problem is that software development experiences dis-economies of scale. As scope increases projects get bigger, requirements for different areas come into conflict, prioritisation gets more difficult, and as more people work on the development the interactions increases.

As a result costs rise disproportionally. A small, highly focused, team will be able to develop a bespoke piece of software for a customer far faster than a large team developing a platform.

Building a platform of any shape or size is significantly more expensive than building a product because it involves working at one or more steps removed from a usable

product. More than one software company has found itself building a platform almost by accident, sometimes because the engineers decide it is a *good thing*.

Software developers love building "platforms" and without strong Product Management input will create a bigger platform than needed. Since building platforms is more expensive this can lengthen development and delay release. But blocking developers' wishes can lead to discontented and frustrated staff.

Aside from code and test there are other costs that need to be considered when embarking on a platform strategy. While it is necessary for a platform to be reusuable it is not sufficient. The easier it is to reuse the platform the more it will be reused, this means supporting the eco-system using the platform. Providing such support costs: documentation needs to be written, training courses delivered, etc. etc.

Some platforms - like Volkswagen's - are intended for internal use, by the companies own engineers. Others, like Microsoft Windows are intended for external use, but engineers outside the originating company. When the platform is intended for external more support, consequently the costs will be higher.

Nor is it enough to offer a well supported platform. There need to be incentives why engineers would choose to develop on a platform. The platform needs to give them something they cannot get otherwise, or make their work much easier.

Offering a platform - even just an API - for external use implies a commitment to future support - few customers will want to buy the product if you refuse to support customisations in future versions. Supporting APIs can be expensive and restricts future options.

Even in the car industry there are some diseconomies of scale: one consequence of VWs extensive use of a few platforms is that introducing a new platform is very expensive and disruptive. When a new platform replaces PQ35 VW will need to redesign every car that uses the platform, retool multiple factories, and market launch many new cars before the full economies of scale are seen. Such high cyclical costs has resulted in some moves to increase the number of future platforms VW uses.

Software development has had significant success with software frameworks that provide tools for developers working within the framework model. Microsoft's MFC was an early success in this field, the Open Source Hibernate framework is a more recent example. However, for every successful framework there are multiple failures.

As well as technical considerations steering developers towards product lines, platforms and framework there are equally powerful marketing factors. Marketing a product as a platform entails creating an ecosystems of application developers before any benefit is seen. It may be better to market the product as a application in its own right and only later reveal the platform capabilities, as Apple did with the iPhone.

While the iPhone is usable as a phone straight out of the box the SAP application suite is not. The entire SAP application suite is customisable and configurable through options, options tables and programming in ABAP (a Cobol like language). This flexibility has allowed SAP to become the dominant enterprise application suite in ERP, CRM, finance and other business areas.

SAP's customisability is a double edged sword, it is so customisable that it is virtually unusable in its uncustomised version. Customising SAP systems is a major development exercise in its own right and provides opportunities for a VALUE ADDED

RESELLER. Consequently there is a whole industry of SAP consultants, designers, developers and VARs.

Product line engineering, platform strategies and the development of product ecosystems are massive topics in their own right. Much has been written elsewhere about these topics at the time of writing, Dietmar Schütz, Markus Völter and others are developing patterns for Product Line Engineering. Outside of the pattern form *Platform Leadership* (Cusumano and Gawer, 2002) discusses the development platform eco-systems.

Providing customisation points

Many of the techniques used to make products customisable are the same as used by designers of platforms, product lines and frameworks. Some are brute force - change the code and recompile, add an if-statement or a compiler macro. Others are user accessible - configuration options, user preferences or macro-scripting languages.

Difficulties set in as increasing variations and options leads to massively increasing code complexity. There are a variety of mechanisms, or "hooks", that can be used together or individually to cleanly provide additional customisability (Figure 6).

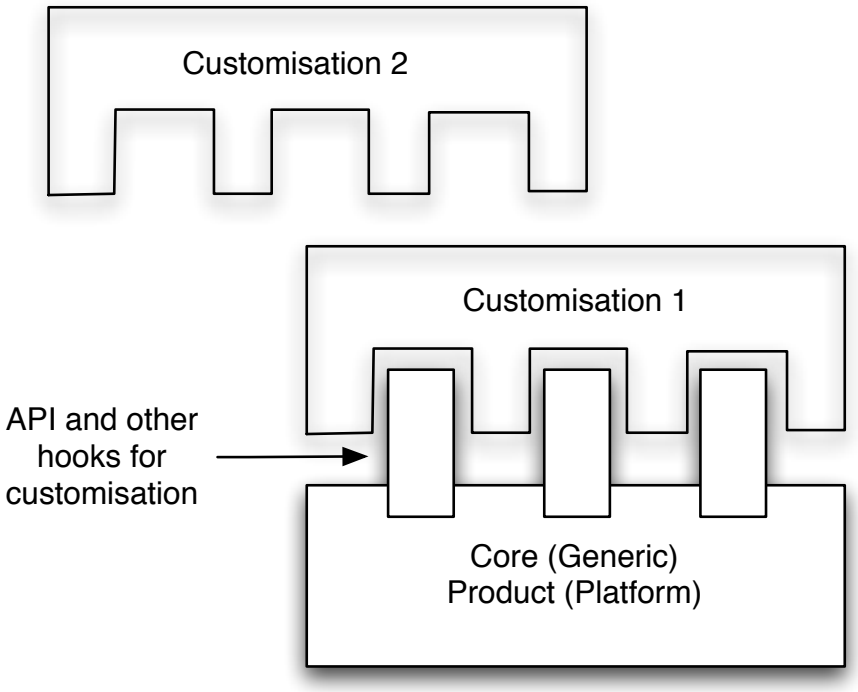

Figure 6 - Customisations can connect to the core product through hooks

Options include:

- Allowing additional plug in features and algorithms using Patterns for Plug-Ins (Marquardt, 2006).
- APIs which allow outside application to drive the product or enhance functionality.

- Offering interfaces which allow external scripting languages to drive the product and use it as a toolbox.
- Building scripting languages into the product which can be used to build functionality or control workflow.

Retrofitting hooks and variation points to an application to allow customisations - let alone turning it into platform - is expensive and technically difficult. However, building a platform from the start is expensive and high risk. Start-up and small companies tend to take the former approach while well-resourced companies take the latter.

While it would be nice to say one approach is superior both approaches have benefits and disadvantages. The choice between one or the other probably has more to do with corporate cash flow than with architectural superiority.

As well as providing the hooks for customisation documentation, examples and support for those customising the product needs to be provided. Some degree of forward compatibility is also necessary so that customisations can be used with future versions of the product.

While it is easy to add a function to an API, it is hard to remove one later. Once added the customisation points will need to be supported in future versions, this will entail testing and documentation updates. Limiting additions to the minimal required and setting a clear retirement policy is needed.

4 Thumbnails for other patterns

CORE PRODUCT ONLY	How do you reduce costs to compete against established companies and products? Only sell the core product, don't bundle anything that is not absolutely necessary. Drive down costs by stripping out non-essential elements and pass these savings onto customers as lower prices. (Kelly, 2005b)
CORPORATE CERTIFIED EXPERTS	You and your customers want to know who is competent to work with the product in depth. Your staff can't do all the work. Therefore, segment the user base by offering to certify those experts who know the product in depth. (Kelly, 2006a)
CUSTOMER CO-CREATED PRODUCT	How do you ensure your product does what your customer(s) wants? Enrol one or more customers as partners in your development process. Give them an opportunity to influence the product design and implementation. (Kelly, 2007b)
EXPEDITIONARY MARKETING	How can you determine which features will be most attractive to the customer when customers are not familiar with your product or how they use it? Launch a product into the market at the first opportunity. Watch customers reaction and how they use the product closely. Use this feedback to develop the next version of the product. (Kelly, 2004b)
HOMOGENOUS CUSTOMERS	How do I enter a market quickly when I do not understand the potential customers? Assume all customers are similar, do not attempt to differentiate or segment the market. Produce one product only. Get the product into the market as quickly as possible while keeping costs down. Treat all customers alike; their money is all good. (Kelly, 2007b)
LIFETIME SERVICES FOR PRODUCTS	How do you ensure your customer get the best experience from your product and you grow your company at the same time? Provide services to your customer over the lifetime of the product so that your objectives are mutually compatible. Your customers will get more from the product; you will get more work and more revenue. (Kelly, 2005a)

POACHER TURNED GAME KEEPER	How do you get an in-depth understanding of your customer, their needs and the pressures they work under? Employ (ex)customers who have experience and knowledge of the application domain you are targeting. (Kelly, 2007b)
PRODUCT ROADMAP	Customer and co-workers need to know what will be in future versions of the product. But you need to be able to change what features are included and when. So create a roadmap that shows future features with approximate dates. Use the roadmap to solicit views and revise the roadmap. Keep the roadmap as a living document. (Kelly, 2008)
PRODUCTS WITH SERVICES	How do you help your customers get the maximum benefit from your products? Supplement your product with services. Start with basic services like technical support and training; later you can expand this with additional services. Good services complement the product and enhance the ownership experience. In the best cases, it is hard to tell were product ends and service begins. (Kelly, 2006a) (rewritten)
SALES/TECHNICAL DOUBLE ACT	How do your avoid overwhelming your account managers with commercial and technical issues? - Both before the sale and the after. Have your customer account managers work in pairs, one handles the commercial aspects of the product and the other handles the technical aspects. (Kelly, 2007a)
SAME CUSTOMER, DIFFERENT PRODUCT	It is easier to sell to existing customer than it is to find and sell to new customers. Therefore have additional products you can sell to your existing customers. (Kelly, 2007b)
SEGMENTED CUSTOMERS	Your customers all seem to want different things, how do you know what features to provide? What documentation to write? And services to offer? Segment your customers into different groups and address the needs of each group separately. Groups are defined on discernable attributes and characteristics that allow you to differentiate one group from another. Working with definable groups avoids generalisations that do not accurately describe any one group.(Kelly, 2007b)
SELF-SERVICE	How do you provide customer service without costs running away as you grow?

	Build a service process that can be operated by customers and allows customers to serve themselves. (Kelly, 2005b)
SIMPLE PRODUCT VARIATIONS	How do you increase variety and differentiation in your products different without increasing costs? Offer a basic product with several simple variations, keeping the variations simple will keep them cheap. (Kelly, 2005b)
SIMPLER PRODUCTS	What new product can you introduce to a market that is already served by an established vendors with existing products? Create a product that is massively simpler to use then existing products and target a different set of customers. Target customers who do not need all the features of the existing products, do not have time to spend learning the product and who find the complexities of existing products off putting. (Kelly, 2007b)
VALUE ADDED RESELLER	Work with one or more partners who can reach the customers you cannot, and allow them to enhance the product as needed. They may add services you cannot provide - an additional line of support, or support in the native language of customers, or professional services to help customers use the product or integrate it into their existing systems. (Kelly, 2009)
WHOLE PRODUCT	How do you ensure technical products solve customer problems and deliver value to customers? Identify a market where your technology can deliver benefits to customers and seek to solve customer problems completely. Match your technology to your market, bundle your generic product with the additional products and services that are needed for customers to recognize the promised value from the product. Market the package to customers in your chosen market as a complete solution. (Kelly, 2008)

Table 1 - Thumbnails for marketing patterns

5 Acknowledgements & Contact

These patterns were presented to a writers workshop at EuroPLoP 2011 at Kloster Irsee, Germany in July 2011.

The author would like to thank Michael Weiss for returning as shepherd for these last few patterns. In addition the author would like to thank the member of EuroPLoP workshop A for their comments and suggestions for improvement: Claudius Link, Anne Hoffman, Neil Harrison, Aliaksandr Birukou, Christoph Hannebauer, Elissaveta Gourova, Yanka Todorova and Lise Hvatum.

Figure 2 and Figure 5 licenses from iStockPhoto.

The author can be contacted at: allan@allankelly.net. More patterns in this sequence are available at http://www.allankelly.net/patterns/business.html.

6 References

CAGAN, M. 2008. *Inspired: How to Create Products Customers Love*, SVPG Press.

COOPER, A. 2004. *The Inmates Are Running the Asylum*, Que.

COPLIEN, J. O. & HARRISON, N. B. 2004. *Organizational Patterns of Agile Software Development,* Upper Saddle River, NJ, Pearson Prentice Hall.

CUSUMANO, M. A. & GAWER, A. 2002. *Platform Leadership: how Intel, Microsoft and Cisco drive industry innovation,* Boston, Harvard Business School Publishing.

KELLY, A. 2004a. Business Strategy Design Patterns. *In:* EuroPLoP 2003, July 2004 2004a Irsee, Germany. UVK Universitassverlag Konstanz GmbH.

KELLY, A. 2004b. Business Strategy Patterns for the Innovative Company. *In:* VikingPLoP 2004, 2004b Uppsala, Sweden.

KELLY, A. 2005a. Business Strategy Patterns for Technology Companies. *In:* VikingPLoP 2005, 2005a Espoo, Finland.

KELLY, A. 2005b. A few more business patterns. *In:* LONGSHAW, A. & ZDUN, W., eds. EuroPLoP 2005, 2005 2005b Irsee, Germany. UVK Universitassverlag Konstanz GmbH.

KELLY, A. 2006a. Patterns for Technology Companies. *In:* HVATUM, L. & ZDUN, W., eds. EuroPLoP, 2006a Irsee, Germany. UVK Universitassverlag Konstanz GmbH.

KELLY, A. 2006b. Positioning Business Patterns. *In:* ZDUN, W. & HVATUM, L., eds. EuroPLoP, 2006b Irsee, Germany. UVK Universitassverlag Konstanz GmbH.

KELLY, A. 2007a. More patterns for Technology Companies. *In:* VikingPLoP 2007, 2007a Bergen, Norway.

KELLY, A. 2007b. More patterns for Technology Companies. *In:* HVATUM, L. & SCHÜMMER, T., eds. EuroPLoP, 2007b Irsee, Germany. UVK Universitassverlag Konstanz GmbH.

KELLY, A. 2008. Business Strategy Patterns for Product Development. *In:* EuroPLoP (European conference on Pattern Languages of Program design), 2008 Iresee, Germany.

KELLY, A. 2009. A Pattern vocabulary for product distribution. *In:* KELLY, A. & WEISS, M., eds. EuroPLoP, 2009 Irsee, Germany. Lulu & CEUR.

KELLY, A. 2012. *Business Patterns for Software Developers,* Chichester, John Wiley & Sons.

MARQUARDT, K. 2006. Patterns for Plug-Ins. *In:* MANOLESCU, D.-A., VOELTER, M. & NOBEL, J. (eds.) *Pattern Languages of Program Design.* Boston, MA: Addison-Wesley.

PINE, B. J. 1993. *Mass customization : the new frontier in business competition,* Boston, Mass., Harvard Business School Press.

RISING, L. 2002. *Customer Interaction Patterns* [Online]. Available: http://members.cox.net/risingl1/articles/customer.doc [Accessed 2 August 2004 2004].

STULL, C., MYERS, P. & SCOTT, D. M. 2008. *Tuned in : uncover the extraordinary opportunities that lead to business breakthroughs,* Hoboken, N.J., J. Wiley & Sons.

WOMACK, J. P. & JONES, D. T. 2005. *Lean Solutions,* London, Simon & Schuster.