The Sun also Sets: Ending the Life of a Software Product

Slinger Jansen[1], Karl Michael Popp[2], Peter Buxmann[3]

[1] Utrecht University, Utrecht, the Netherlands
s.jansen@cs.uu.nl
[2] SAP AG, Waldorf, Germany
karl.michael.popp@sap.com
[3] Darmstadt University of Technology, Darmstadt, Germany
buxmann@is.tu-darmstadt.de

Abstract. Sunsetting a software product is a painful and frustrating process, whether it happens in times of crisis or in an organized and planned manner. It is surprising that little information is available on how to perform sunsetting and it appears to be a blind spot in software product management literature. This paper describes the sunsetting method and provides practitioners with a well-defined process of how software products must be taken out of development, maintenance, and finally use. With the sunsetting method, product managers will have as little trouble as possible based on the experiences of others. The process description is elaborated using a method description. Furthermore, three retrospective case studies have been conducted to evaluate the method.

Key words: sunsetting, product software, end-of-life, method engineering

1 Introduction

We define sunsetting as the process of planning and executing the end-of-life of a software product that is currently in use by customers and maintained by a software producing organization. The end-of-life of a software product describes the point after which the product is no longer maintained or supported by the manufacturer of the software product. Phase-out is an alternative term for sunsetting. Sunsetting is actually part of the portfolio management process.

Portfolio management is defined as the strategic decision process, whereby a business portfolio is constantly updated and revised in order to meet business objectives [1]. In the context of software vendors, the aim of portfolio management is to get the most out of a companys investments and products [2]. Good portfolio management requires close oversight, constant review of historical and current performance, and the courage to rebalance and rationalize the portfolio when necessary while aligning your actions with the overall strategy of the organization [3]. Basically, product phase-out is part of the continuous assessment that a software product manager undertakes when evaluating the product portfolio.

In a time of double-digit growth in the software industry, it may seem illogical to address the death of a software product. Why would one wish to phase out an unsuccessful product when there is always the chance that a customer might order it, or orders extra licenses out of the blue? As more experienced software product and portfolio managers

know, there are many reasons to do so. These reasons are found in three categories, being product strategy, platform changes, and portfolio decisions [4, 5].

In regards to product strategy, there are simple reasons such as release deprecation (e.g. a software vendor only supports the last two minor releases) and a lack of demand for the product. Another reason may be that maintenance becomes too expensive, i.e., upkeep for multiple products is too expensive for one company, or when developers for a technical platform become too scarce. It must be noted that a products life expectancy and potential profitability is more relevant than current profitability. If a product is successful presently, but will be hard to monetize in a couple of years because the product is no longer needed or based on technology that will become outdated soon, it can become a candidate for discontinuation.

The platform on which the product is built can also change the course of a products lifecycle. If a new release is made of the underlying technology, for instance the operating system as a basis for applications, the product owner has to decide when the releases of the product based on the older platform are no longer required to maintain a healthy business. Another platform on which the product depends might be a database system. If a database system is phased out, the product needs to evolve as well, or be phased out as well.

Finally, there may be portfolio decisions that end the life of a product. A product may be an inferior duplicate to another product, or a product may be outdated. Another reason may be that the product is no longer profitable or even performing badly in such a manner that it is causing harm to the software vendors reputation. Finally, legal constraints may force a product owner to kill off a product. These legal constraints may be that the company has formed a monopoly and is forced to reduce specific activities, or that intellectual property laws are broken by the product.

There are several factors that influence the ideal moment, with the least damage to the company, to end the life of a software product. Environmental influences, such as the entrance of a new standard or the introduction of regulatory requirements, such as XBRL-based reporting (a universal standard that allows for automatic processing of business accounting data), may mark such an ideal moment. Also, please note that there is a difference between ending the life of a product all-together and ending its life as a customer of an application [6], even though these processes have many things in common, such as changing customer needs, technical change, regulatory change, and competitive change.

To further illustrate, we take the example of Microsoft when it ends the life of one of its own products, by taking a closer look at one of the Windows versions. Microsoft publishes three dates to customers for each version of the Windows operating system in regards to sales, being the date of general availability, retail end of sales, and the end of sales for PCs with Windows version pre-installed. Furthermore, three dates are published in regards to support, being the publication of the latest service pack, the end of mainstream support date, and the end of extended (paid) support date. Obviously, for some organizations a major operating system upgrade is a huge undertaking, in terms of system maintenance (imagine an organization with over 5000 workstations), in terms of system compatibility (organizations easily have over 10,000s of applications of which many are compatible with one version of Windows only), and in terms of in-

vestment (hardware may be outdated, the upgrade will involve acquiring new licenses). An interesting detail of Microsoft's terms of service is that pre-installed deployments of Windows sometimes have downgrade rights, enabling the customer to downgrade to a previous version of Windows that is compatible with the other Windows versions in the organization.

We continue this paper by describing the research method in section 3. The research method is followed by a decomposition of the interactions between software vendor and customer in section 2, to illustrate what type of agreements need to be dismantled when ending the life of a software product. In section 4 the product software discontinuation method is presented and described in detail. Section 5 continues with the description of three case studies that illustrate the method and show the intricacies of the sunsetting process. Finally, in section 6, the conclusions are derived and discussed.

2 Decomposing Sunsetting

We now provide a further explanation of the complex operation of sunsetting. This paper looks specifically at on-premise software, which is provided from a software vendor to the customer. For the sake of simplification, let us assume a simple, direct relationship between the software vendor and the customer. In this case, sunsetting is like rolling back a distributed transaction between the software vendor and a customer. This distributed transaction can be divided into three subtransactions: *Provide software*, *Provide maintenance* and *Provide support*. While rolling back this transaction and its three subtransactions seem simple, the next level of detail shows that there are subtransactions that cannot be rolled back. They need compensating transactions and thus introduce complexity and efforts into the process of sunsetting solutions. Another factor for adding complexity and efforts is customer lock-in.

Provide software - The transaction *Provide software* is divided into the subtransactions *Provide a copy of the software*, *Transfer usage rights* and *Provide license key*. The first transaction *Provide a copy* of the software can be easily rolled back if the customer has a time limited license. The customer just has to give back the copy of the software at the end of the license term. If the customer has a perpetual license, the customer can keep the copy of the software. The second transaction *Transfer usage rights* cannot be rolled back in a simple manner. Based on the contract terms, the customer can keep the usage rights, the usage rights can end. If the customer has perpetual usage rights, he needs to get usage rights on a software product that replaces the sunsetted product.

Replacing the sunsetted software product introduces more complexity and effort for the software vendor and the customer. Replacing a sunsetted software product with a new software product means that the results of activities that have gone into installing, implementing, maintaining and running the software cannot be rolled back and usually carry high sunk cost. A compensating transaction has to collect the results of these activities and migrate these results into a new software product that replaces the sunsetted product. Simple examples for these results are customer data or customer specific extensions of the software product. How this replacement is properly planned and executed will be covered later in this paper. The third transaction *Provide license key* follows the same logic of rolling back as *Transfer usage* rights.

Provide maintenance - *Provide maintenance* is divided into subtransactions *Provide new release*, *Provide new version* and *Provide bugfix*. Over time, the customer is served by multiple executions of these subtransactions. At a certain point in time, the customer arrives at a combination of release, version and bugfix. In the case of replacement of sunsetted product, the customer needs a replacement of exactly the combination of release, version and bugfix he is currently running. This shows additional complexity in replacing sunsetted software, since each of the customers might have a specific combination that might differ from the combination each other customers have. Numerous instances of these transactions have been executed and have lead to the current system landscape at the customer.

Provide support - The third high-level transaction is *Provide support*, which aims at providing resolutions or workarounds for customer issues with the software. Each of the transactions was executed several times. The result of the transactions is a customer specific set of resolutions or workarounds. The transactions do not have to be rolled back.

3 Research Method

The research question of this paper is:

How can a method be created for a product manager to sunset a product (line)?

The research question is answered by applying method engineering in a design research project. Method engineering is used for designing, constructing and adapting methods, techniques and tools to develop information systems [7]. Design science is an outcome based information technology research method, which offers specific guidelines for evaluation and iteration within research projects [8].

Research Execution - The research consists of three steps. A first version of the method was created to create a baseline method, based on literature and experience. The method is evaluated with several experts from the industry (with 15, 15, and 13 years of experience), who have long-standing experience with retiring and sunsetting software products and product lines. Thirdly, the method is evaluated by doing three exploratory case studies, to establish that the method is complete. The case studies are listed in table 1 and further discussed in section 5.

Method Engineering - van de Weerd et al. [9] describe a meta-modeling technique based on UML. This technique depicts a method in a Process-Deliverable Diagram (PDD). A PDD consists of two parts: a process model (UML activity model) on the left side and the deliverable model (UML class diagram) on the right side. An example PDD is depicted in Figure 1, which models a highly simplified requirements engineering process. On the left-hand side an activity called “Requirements elicitation” is modeled, which contains the sub-activity “Write requirements document”. The requirements engineer executes the activity. The activity results in a deliverable called “Requirements Document”, which has several properties. The main reason for using this type of method descriptions is that it enables us to present the sunsetting method in a structured manner, and enriches the main contribution of this paper from a simple

checklist, to a rich method description that can be reused by practitioners and improved upon by academics.

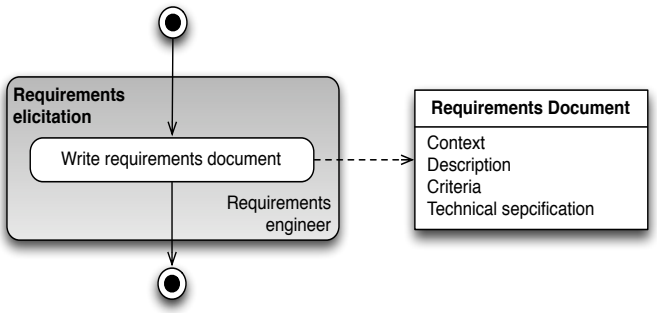

Fig. 1. Process-data diagram [9]

Case Study Company		**Phased out product**				
Identifier	**# of employees/products**	**Age**	**Location**	**Reasons for phase-out**	**# of employees in product unit**	**Age**
PubComp	18000/40+	15*	Netherlands, Europe	Out of portfolio scope, not making targets	120	20
ERPComp	51000/100+	38	Germany, world-wide	Redundant, rebranded product branch, contract standardization, reduce maintenance efforts	20	10
ServicesComp	9100/100+	18	Netherlands, Europe	Duplicate functionality, aged technology	135	16

Table 1. Companies and Products (* age of the software business, the company was established in 1878)

The three companies were opportunistically selected, since we had access to board level management in each of the companies. Each of the companies, however, has a long experience dealing with large product portfolios and were selected with that reason in mind. Each of the interviewees was working at one of the case companies at the time. The interviews were undertaken in four steps. First, a general discussion was had about the organization. Secondly, we discussed the topic of phase-outs, and tried to establish the interviewees view on the topic, including any previous experiences the interviewee had with the topic. The interviewees were asked to develop a quick outline for a method themselves, to see whether they understood the topic and to see whether their experiences further confirmed the first version of the method. Documentation, if

available, was handed over in regards to product end-of-life. In the third step, the first version of the method was shown to the interviewees and walked through with the exact same narration for all interviews. The interviewees were allowed to comment on the method and all three at some point grabbed a pen to make their own additions and changes. During step four, an example from the interviewee's past was taken to see whether the method was followed in any way.

Three interviews were undertaken, of which two in Dutch and one in English. The method was always described in English and each term was explained in a glossary, which the researchers brought to the interview. Interviews took between 100 and 150 minutes. Each of the interviews was recorded. The interviews were conducted by one researcher only. The results from one of the interviews were checked by a second researcher.

4 The Product Software Discontinuation Method

This section describes the method shown in figure 2. On the left side the method activities are displayed, on the right side the deliverables created during the execution of the method are shown. These deliverables are further explained in table 2. The first version of this method was created from literature, the second version was created based on three interviews and case studies.

The first version of the method was in part inspired by IEEE std 1074 [10], a process standard for the software lifecycle, which provides a concise description of the retirement process. The description consists of three steps, being "notify user", "conduct parallel operations", and "retire system". The "parallel operations" step consists of using two systems simultaneously, while one of the two is being phased out. The IEEE standard provides some insight into the retirement process, but does not specifically address the challenges a software vendor may experience during a product phase out. The method was also inspired by several product phase-out overviews from larger software vendors, such as the Microsoft Windows phase out web pages, the information pages from SAP about Business Object's (acquired by SAP) SRC, and the pages from Cisco about the Quality of Service (QoS) Device Manager Software, which was phased out over a long period in the previous decade.

Discontinuation Assessment - Any organization that maintains a software product must regularly assess the viability of its products and product lines, as part of the portfolio management process. This process consists of reviewing the portfolio plans for current products, assessing their success, profitability, market size, and growth. When a product becomes a potential candidate for discontinuation, a customer assessment needs to be done to establish how important the product is for these customers and how important these customers are, since discontinuation of the product could mean the termination of a long-lasting relationship. Finally, a list must be created of all the products that depend on the product that may be discontinued. In case of discontinuation, the teams behind all products on the list of dependent products must be informed.

Phase-out Planning - Whenever a phase-out is impending, software vendors need to evaluate possible alternatives before actually phasing out a software product. There are several alternatives to phasing out a software product that still ensure continuation of

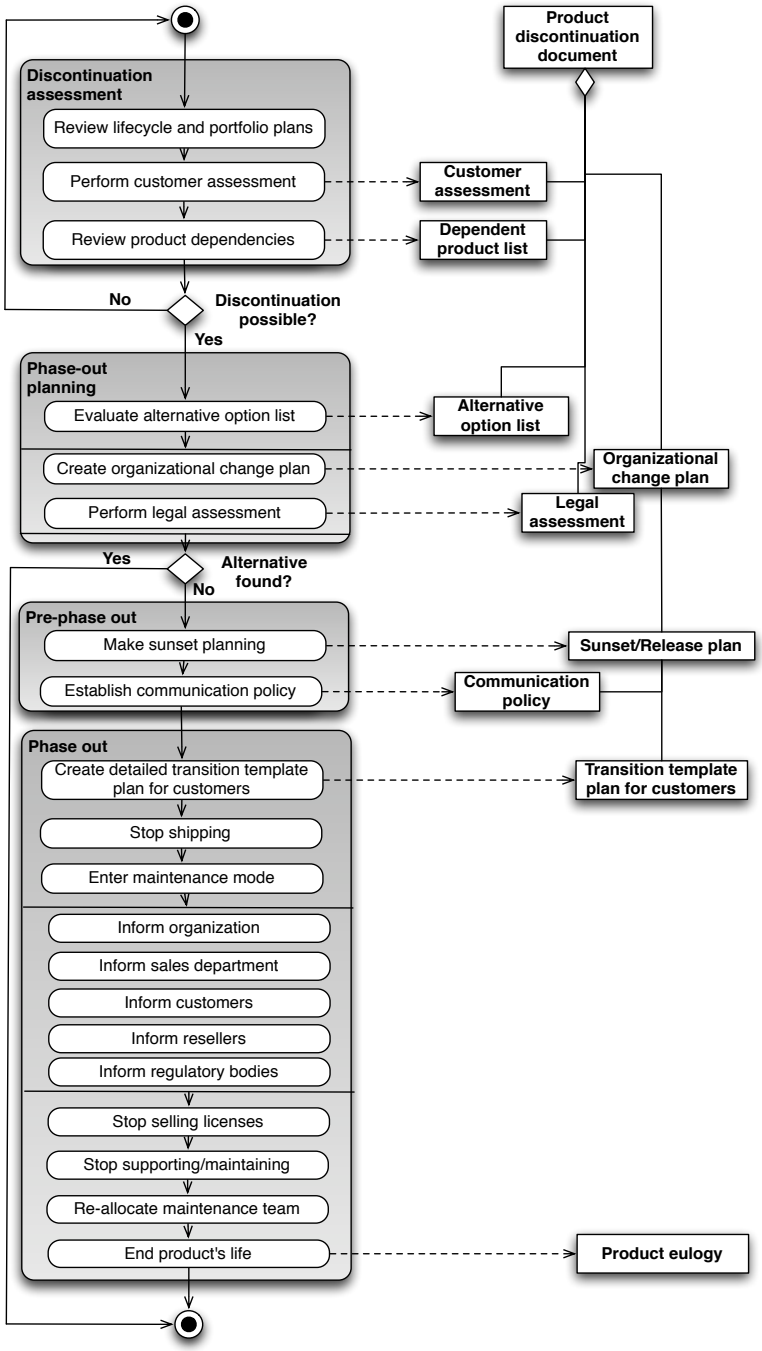

Fig. 2. Product Software Discontinuation Method (all activities executed by product owner)

the business, although perhaps with different levels of quality of support. The following list is not exhaustive, but presents some of the alternatives to completely phasing out the software product:

- **Open source -** it is possible to establish an open source community that further supports and maintains the product. This option, if the product is not already open source, should provide a real alternative, i.e., a sustainable community must be created that maintains and provides support, and there should not be a contractual conflict in regards to licenses.
- **Management Buy-out -** if a product is being maintained by an isolated group of people within the organization and the product could become more profitable if it did not have to support the parent organization, one option to avoid product phase-out is to have a group of adequate managers or key people in the product group buy the product out of the portfolio, and continue as an independent organization.
- **Sale -** when a product no longer fits the portfolio of an organization, it can be sold to a third party, along with (parts of) its maintaining organization. Organizations will generally go to great lengths to sell off a product instead of killing it off. A common phenomenon is that an organization starts "putting lipstick on the pig", i.e., trying to maximize profit and growth numbers to make the product group more attractive to a potential seller.

Based on the alternatives, an organizational change plan is created and a legal assessment is performed. The organizational change plan establishes how the product discontinuation is implemented in the organization, together with a timeline of events. A legal assessment is performed as well, to establish the consequences of the different variations of the plan. The legal implications can be serious, in that contractual obligations towards customers and partners may need to be altered to accommodate a products discontinuation.

Pre-phase out - If no alternatives are found, the product will need to be phased out. To do so, a planning is made that details the steps that are taken in the process. Such steps include the cessation of maintenance and support of a product, and the last sale of licenses. One of the most precarious processes of the phase-out is the establishment of a communication policy, due to the nature of the phase-out process. Phasing out a software product entails in some cases the complete disbandment of a products maintaining unit. The impact the discontinuation of a software product can have requires that a communication policy is set-up that minimizes damage to the organization.

Phase-out - After a phase-out plan has been crafted, the plan must be executed, starting with the creation of a customer transition plan template, which lists several routes to a new solution for customers, such as migration to a replacement product. This is a template and must be adjusted for each individual customer, since each customer operates from a unique situation. If the product is still being sold that must be stopped. Furthermore, the product must fully go into maintenance mode, such that no new functionality is added to the product, with the exception of updating time-sensitive content. Once the product is in maintenance mode, the communication policy is executed according to the timeline created earlier. After communicating with all stakeholders that the product will be discontinued, a last round of license sales can be done,

after which license sales are ceased as well. As soon as the product legally no longer requires support and maintenance, these processes can be stopped as well, after which the maintenance team needs to be re-allocated. Finally, the product is finished, and the product manager or entrepreneur who has been responsible for the phase-out process, can write a product eulogy.

Painful Process - The method description does not sufficiently show that the process of phasing out a software product actually has great consequences for the people working with it. Think, for instance, of the support engineer who knows every nook and cranny of the software product, or the user who has configured the product just to her specifications and is described as the wizard of that product by her colleagues. We advise practitioners to make compromises and be sensitive towards the emotions that surround legacy products, both in their internal and external communication. By taking it slow and on-boarding fanatic proponents of the legacy products, transitions may potentially go much easier.

5 Case Studies

The case studies were performed to provide different examples of how the lives of products are ended. The cases served as a measure to evaluate the method provided in figure 2.

5.1 Case Study: Health and Safety Product at PubComp

Context. The HSP (a Health and Safety Product) was a relatively successful product that no longer supported the business goals of PubComp, a software vendor in the Netherlands with a large portfolio. The HSP came up in regular evaluations at PubComp as being out-of-scope of the product portfolio, and was consistently underperforming.

Process. These evaluations were top-level management evaluations, and were treated as top-secret, since they deciding on the future of approximately 120 people. After considerable time, a team was identified within the HSP business unit that could potentially undertake a management buy-out. The HSP team and PubComp agreed to a buy-out price and strategy. Within 18 months the HSP business unit was functioning independently and the formal contract was signed.

Findings. When a management buy-out or the sale of a product to another company is in sight, some interesting processes start taking place. There are issues with personell, resources, and business unit value determination (although one could argue this is relevant for 'regular' phase-outs as well). Purchasers attempt to find all the things that could bring the value of the company down, whereas the seller will be tempted to make the business unit look more successful than it actually is. The sale of a business unit is a more organic and management-directed transition than a planned phase-out.

Impact on the method. The first version of the method assumed that the end of a life of a software product always entails the complete ending of sales, support, etc. This first case already showed that it is rarely the case that the product, whose customers always represent some kind of business value, is completely phased out without

Concept Name	Concept Description
PRODUCT DISCONTINUATION DOCUMENT	This document describes the complete plan on how to discontinue the product. The creation of the document only expresses the intent to explore the options for discontinuation and plans may be discarded after the LEGAL ASSESSMENT and CUSTOMER ASSESSMENT have been completed.
CUSTOMER ASSESSMENT	The CUSTOMER ASSESSMENT is performed to explore what the effect will be for customers and how relationships will be altered after discontinuation. The assessment includes a general assessment and a specific assessment per individual customer.
DEPENDENT PRODUCT LIST	The DEPENDENT PRODUCT LIST lists all products that are dependent on the discontinued product and therefore may also have to be discontinued.
ALTERNATIVE OPTION LIST	The list of alternatives provides an overview of alternatives to discontinuation, such as the sale of the product to a third-party.
LEGAL ASSESSMENT	This document describes the legal risks and consequences, based on the CUSTOMER ASSESSMENT and DEPENDENT PRODUCT LIST.
ORGANIZATIONAL CHANGE PLAN	This document describes how the organization will change in the following period as the product is discontinued.
SUNSET/RELEASE PLAN	The SUNSET/RELEASE PLAN describes how the product was planned to be phased out, if such a plan is available. If not a sunset plan is created. It is also called a lifecycle plan.
COMMUNICATION POLICY	The COMMUNICATION POLICY describes how and especially when the discontinuation plans are communicated. These documents are generally sensitive and must be treated so by the entire organization to avoid leaking.
TRANSITION TEMPLATE FOR CUSTOMERS	Based on the CUSTOMER ASSESSMENT a TRANSITION TEMPLATE is created for each customer group, that can be filled in by a consultant, as to advise a customer in the transition to (for instance) a new system.
PRODUCT EULOGY	The PRODUCT EULOGY describes what the product was, how well it performed, and why it was phased out.

Table 2. Concept Table

any viable alternatives for the product (management buy-out or purchase by another company) or for the customers (in the form of a product alternative, for instance).

5.2 Case Study: Enterprise Resource Planning Product at ERPComp

Context. A large platform provider in Germany has been growing autonomously, for almost 40 years. The company has gone through several product sunsets, but availability of knowledge on these processes within the organization remains scarce. In 2007, ERPComp acquired a company that specializes in identity management products and was aimed to be complementary to the ERP platforms supplied by ERPComp. Throughout the years, ERPComp integrated some of the technological feats that came with the iden-

tity management products into a new identity management platform, thereby making some of the original products redundant. In 2007 it was decided that the last products that were still being supported by ERPComp from the acquired company were to be sunsetted. In total, different versions of six products were to be phased out.

Process. First, an overview was created of the different products that were currently in use and of how many active customers each of these products had. Secondly, an overview was created of upgrade and migration routes that could be traveled by customers. Part of this overview was a plan that described what happened when a customer ordered extra support, licenses, or even a new deployment.

Findings. An extra complication was that some of the products had been resold and rebranded under another label and these also needed to be phased out. Phasing out the rebranded product proved to be a challenge, since the resellers that sold the rebranded product also had service contracts with their customers. As soon as ERPComp phased out the product, the reseller contracts were also ended. Because of this, the customers of those resellers no longer received support and maintenance. ERPComp solved this elegantly by timing exactly when these customers would be willingly transitioned from the reseller to ERPComp.

The phase-out timeline had to be extended twice with one year, because ERPComp wrongly estimated the availability of consultants that would be needed for each transition. Furthermore, for some customers it proved to be interesting to extend the maintenance period further, against much higher license costs. A positive observation was that out of approximately 200 customers, there was no attrition.

Another challenge was that the transition template plan for customers had to be updated, due to the fact that new releases of the replacing platform came out. This had not been taken into account, and introduced extra cost into the phase-out process. ERPComp required support from its ecosystem of partners in phasing out the product. To gain their support, the partners were informed early of the consequences, and communication protocols were established. Partners were also trained in providing support during transitions, such that customers could receive support from their preferred partner at the same quality level as ERPComp would provide.

Impact on the method. The role of the communication policy and legal assessment were further outlined during this case study. ERPComp has many large customers, which expect the absolute best from their ERP vendor, and are willing to pay for it. If ERPComp makes any serious changes, a network of customers exists that is powerful and can litigate if necessary.

5.3 Case Study: Three Municipalities Products at ServicesComp

Context. A large software and services company in the Netherlands has been growing mostly through acquisitions. Three software producing organizations creating products for municipalities were acquired in 1996. Each of these software producing organizations has become a separate business unit of ServicesComp. Two of the products are complementary, whereas the third product copies up to 70% of the functionality of the other two. All three products have been developed on older technological platforms and all three products are currently maintained and marketed separately. ServicesComp perceives that three business units building similar products is inefficient and an initiative

has been started to build a new best-of-breed product that will replace the other three. The decision to reduce the inefficiency has been made a long time ago by the management team of the business division. No structural approach was implemented at that time at ServiceComp for portfolio management, but at the time of acquisition it was already clear that the three products had overlap and that a merge would happen over time.

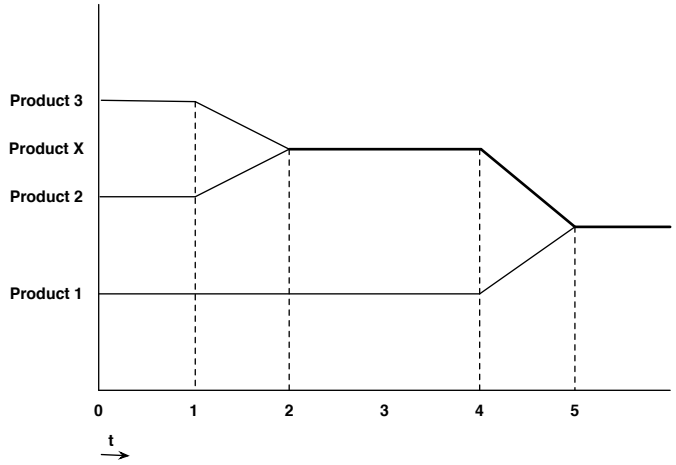

Fig. 3. Transitioning from Three Products to a New Product

Process. The actual merge has been visualized in Figure 3, which shows how the three products are phased out and replaced out by a fourth. The two products that are complimentary are phased out first, the third product will be phased out last. ServicesComp has steady five-year contracts, with yearly extensions after that. It is clear to ServicesComp when the last contract can potentially be ended, so planning for the merger of the products is more dependent on development speed than contractual obligations. ServicesComp looks positively toward the transition of customers from their old products to the new. To begin with, the transition is a business opportunity for ServicesComp, who provide a lot of services, and as their competition has made some mistakes in the past in regards to transition, ServicesComp is unafraid of customers transitioning to another vendor. ServicesComp has indicated that customer research is a major step in a product discontinuation document, since transitions will go much smoother and entail less risk if customers perceive the supplier of the original product positively.

Findings. When it was clear that the functionality of the three products would be merged into product four, the organization prepared a communication protocol. Parts of this protocol were reused to establish the rules of (potential) customer communication for sales personnel. ServiceComp used the following elements in this protocol:

– Roadmap of product X

– Rough timeline for phase-outs
– Generic features, such as connectivity
– News embargo on product 1, since the planning hadnt been finalized

The communication protocol was created to ensure customers that service for the products would be continued and discouraged them to look at competing products by describing the advantages of the new products.

Impact on the method. Again it was confirmed that providing alternative options to customers is a successful way of retaining them as customers. Furthermore, because ServicesComp has a mature view on the phase-out process, the case study functioned as a confirming case to show that the method does not miss any essential steps. None such steps were found and the second version of the method, as shown in this paper, could be published.

6 Discussion and Conclusion

The sunsetting process of a software product that has been deployed in the market is a complex and loaded process. It involves changes to the product portfolio of an organization, an impact analysis on the company, and structural change in an organization. The process should not be taken lightly, or mistakes will be made. In our case studies alone, we found examples of deadlines that were severely delayed, customers that experienced serious financial damage, and missed business opportunities (not offering a replacement product, for instance). The method provided in this paper hopefully helps organizations avoid future problems when ending the life of a software product.

During our research with experts and at companies, it became clear that this topic is considered sensitive. We have looked for method descriptions, process descriptions, and phase out plans, and found that these documents were always confidential, if available at all. We also found that because of its nature, the sunsetting process is generally performed ad hoc, even if a company is highly experienced in the field. It has become apparent that case studies and interviews are the best way to develop the method that has been created during this research.

It may appear that aspects of our approach (take it slow, communicate early, use a structural approach) are directly opposed to the profit goals of an organization. After all, if a business unit can be sold off, the sunsetting process is slow and mechanical, and the seller may feel it is missing out on a great deal. We hypothesize, however, that a structured approach will lead to less problems along the way and may even uncover that the life of the product (or business unit) is not yet over or cannot be over, due to customer lock-in.

In this paper we have focused on ending the life of a software product. It is surprising how little literature is available about such a meticulous and sensitive process, and this paper attempts to fill that void. We have defined the process of sunsetting, devised a method to support the process, and given some hints towards reasons to start sunsetting. We hope that practitioners will benefit from our research and that the scientific community further expands on how and when to sunset a software product.

References

1. Pohl, K., Bckle, G., van der Linden, F.: Software Product Line Engineering. (2005)
2. Popp, K.M., Meyer, R.: Profit from Software Ecosystems. Books on Demand GmbH (2010)
3. Haines, S.: The Product Manager's Desk Reference. McGraw-Hill (2008)
4. Jansen, S., Brinkkemper, S., Finkelstein, A.: Business Network Management as a Survival Strategy : A Tale of Two Software Ecosystems. Proceedings of the First International Workshop on Software Ecosystems (2) (2009) 34–48
5. Weerd, I., Bekkers, W., Brinkkemper, S.: Developing a maturity matrix for software product management. In Tyrvinen, P., Jansen, S., Cusumano, M.A., eds.: Software Business. Volume 51 of Lecture Notes in Business Information Processing., Springer Berlin Heidelberg (2010) 76–89
6. Furneaux, B., Wade, M.: The end of the information system life: a model of is discontinuance. SIGMIS Database **41** (May 2010) 45–69
7. Brinkkemper, S.: Method engineering: engineering of information systems development, methods and tools. Information and Software Technology **38** (1996) 275–280
8. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. MIS Quarterly **28**(1) (2004) 75 – 105
9. van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L.: "on the creation of a reference framework for software product management: Validation and tool support". Proceedings of the 1st International Workshop on Product Management, Minneapolis/St. Paul, Minnesota, USA (2006) 3–12
10. IEEE Standards Board: IEEE Standard for Developing Software Life Cycle Processes, IEEE Std 1074-1997. (1997)