Business Continuity with SaaS

T. van de Zande and S. Jansen

Dept. of Information and Computing Sciences,
Universiteit Utrecht

Abstract. Organizations are increasingly adopting SaaS-solutions in favor of traditional on-premise solutions, because of the clear advantages in terms of cost reduction, implementation time and scalability. Business continuity of these SaaS-solutions is often neglected, even when business processes that depend on the SaaS-solution are critical. This paper addresses business continuity for SaaS-solutions by identifying and evaluating different business continuity solutions to protect customers for the risk of their SaaS-provider facing financial problems or even bankruptcy. Two solutions; 'SaaS-escrow' and the 'SaaS-guarantee-fund', are evaluated in several expert interviews and a questionnaire.

1 Introduction

Software as a service (SaaS) is a form of software deployment, that delivers software on a subscription basis through the internet. With SaaS, companies no longer have to buy or develop a complete software solution up front, instead they "rent" it. Pricing models can vary, but most often customers pay on a subscription-basis or on a usage volume-basis [1]. SaaS is rapidly growing in popularity. Recent developments in Internet technology and broadband adoption have made it possible for online software to serve as a true desktop software replacement. Also the recent economic and financial crises have played their part in showing the advantages of SaaS, because of its low upfront investment and scalability. Because of these advantages in comparison to traditional software licensing models, a large number of small businesses and start-ups are already using SaaS solutions for some time now, and even large institutions and corporations are migrating their data from on-premises software to externally developed and hosted online software.

Service Level Agreements are legal documents where customer and provider agree on the quality of service, by quantifying minimum quality of service [2]. For SaaS, these are usually things like uptime, availability and response time. There are some problems that are generally neglected, like what happens when a SaaS-provider goes out of business. If a company is using mission-critical software that is hosted off-premises, it could get into big problems when the provider decides to pull the plug. For example; what do you do as a retail company when your online retail system goes out of business? Or what does a sales representative do when the CRM system goes offline? These problems can be a deal breaker for a lot of decision makers, so it is clear that there needs to be a solution to assure continuity before SaaS can truly serve as a replacement for offline software. This is not the first time that software delivered as a service receives attention from business clients. In the late 90's of the previous century, application service providers started hosting applications off-site, offering some of the same advantages that current SaaS-providers offer. In the end those traditional ASPs were not able to deliver reliability and quality standards demanded by their business customers [3]. The business models of most traditional ASPs were fundamentally different from current SaaS business models; traditional ASPs repackaged existing legacy software and offered it off-site, so customers still had to buy a license to use the software [4, **?**]. After the burst of the "dot-com bubble" in the year 2000 a lot of those traditional ASPs, like many other internet companies, faced financial problems, and some even bankruptcy, leaving their clients with no access to their software [5, 6]. The new generation of ASPs who offer true SaaS-solutions differ in a way that they do not resell existing enterprise applications, instead they develop their own web-based applications on a new multi-tenancy design paradigm, which makes serving multiple clients more scalable and cost-effective [4].

With shrink-wrapped- or product-software, business continuity problems are usually solved by source-code escrow. Source-code escrow is a legal agreement that protects customers by allowing them to obtain source code and other information that is needed to continue support and development of critical software, when the original developer becomes unable to maintain the application or when it goes out of business [7]. The developer deposits this information with a trusted third party (an escrow company), and only when predetermined conditions are met, the escrow company will release the source code to the customer. This way the customer is assured that maintenance and development for the software can continue, while the developer does not have to release its source code while the software developer is still in business.

For SaaS solutions, this type of escrow does not work. The difference is that for original software, the system would continue to work even if the developer stops supporting the application. The only problem is that there is no maintenance and that there are no updates. The customer has time to find a new developer and the new developer has time to study the source code while continuing operations. With SaaS solutions on the other hand, the developer is responsible for the hosting. When they go out of business for whatever reason, the customer almost immediately does not have access to his software anymore. If the customer arranged a normal escrow solution, it would take a minimum of several weeks before the system could be up and running again, and the costs involved would be much higher than what was initially paid for the SaaS solution.

So to take SaaS to the next level, as a more cost effective and flexible alternative to shrink-wrapped software, it is imperative that there is a clear way for both customers and providers to arrange a viable continuity solution. The order of this paper is as follows; In Sect. 3 we will first discuss how shrink-wrapped software customers are currently protected. Section 4 will discuss what a successful solution should look like for SaaS continuity. In Sect. 5 we discuss what solutions are currently being offered and discuss the necessity of business coninuity solutions for SaaS in general. In Sect. 6 we discuss the results of the survey. In Sect. 7 we conclude the paper, by stating that most SaaS-customers do not need a complete business continuity solution. For some specific scenario it can be important to arrange business continuity solutions. Which continuity solution works best depends on the specific SaaS-solution.

2 Research Method

The goal of this paper is to give a *state-of-the-art* report on continuity solutions for software-as-a-service. We will discuss the necessity of such continuity solutions and compare different initiatives. Since there is no clear solution to the aforementioned problem, the research is explorative. First we will conduct a literature-study of existing solutions for shrink-wrapped software. Then we identify the requirements needed for a successful SaaS solution. We identify these requirements using a survey with business experts and prospective customers. For a more in-depth view of different problems and solutions, several semi-structured interviews are conducted. We also study existing SaaS continuity initiatives and companies who offer such a solution. Finally we compare these existing solutions with the requirements we identified, and discuss the necessity of business continuity solutions.

Name	**Product**	**Customers**
Celerity ICT	SaaS CRM	??
Plan8	SaaS Planning suite (MijnRooster)	39
AllSolutions	SaaS ERP software	80
Escrow4All	Saas-Escrow	20

Table 1. An overview of the interviewed companies along with the SaaS-product they offer and the number of customers they have for that SaaS-product.

3 Source-code Escrow

A source-code escrow arrangement is an arrangement between a software vendor, the buyer and a trusted third party. The software vendor and buyer get into an escrow agreement for several reasons. One of the most common reasons for the buyer of a software license, or licensee, is to mitigate the risk of losing support and maintenance for his software when the developer/vendor, or licensor, fails to do so [8]. On the other hand it protects the licensor from unauthorized access to his intellectual property, because the licensee only receives the source code after certain agreed upon release conditions are met. To ensure the interest of both parties a trusted third party, the escrow-agent, receives and stores the source code of the licensor. The escrow-agent ensures that the code is only released to the licensee when the licensor fails to meet certain release conditions. Over the years source-code escrow has become the favorite solution to guarantee continuing support and maintenance for regular, licensed software [9].

A typical three-party escrow solution works as follows; when the agreement is in place the licensor will deposit a copy of the source code and other documents that are necessary to compile and maintain the software with the escrow-agent. The escrow-agent checks the code for errors and, depending on the contract, can compile and test the software after each update to ensure that the licensor deposited complete and working code[1]. The escrow-agent then stores the deposited code. Usually the licensor will update the escrowed code semi-annually or after big software updates. If at some point the licensor defaults on its support or maintenance obligations, or fails to meet other goals stated in the escrow contract, the licensee can send a request to the escrow-agent to release the stored source-code. The escrow-agent then checks if the licensor indeed violated the escrow contract and sends a notice to the licensor that a release request has been made. When the licensor does not respond to this notice in a timely matter, and still violates the contract, the source code will be released to the licensee. When the licensee gains access to the source code, he can try to find new developers willing to work on the software.

3.1 Is Source-code Escrow useful?

The scenario described in 3 is very simplified and stylized. In reality, most escrowed source-code is never actually released [7]. In many cases the licensor just did not get into trouble, so a release request was never filed, but in some cases the source code was not released even though the licensee requested it. For example when the licensor simply does not approve the release of the source-code [10]. A good example of this is a recent court case between Vemics, Inc. and Radvision, Ltd. (Vemics, Inc. v. Radvision, Ltd., No. 07-CV-0035, 2007 WL 1459290 (S.D.N.Y. May 16, 2007)). In this case the licensee, Vemics Inc., requested the release of the source-code from the escrow-agent, Iron Mountain, because the licensor, FVC, filed for bankruptcy. However Radvision Ltd., the successor in interest to FVC's rights and obligations, obstructed the release because it did not agree that a valid release condition had occurred. as of this day, it is still not clear if the source-code will ever be released [11]. Such examples raise questions on why source-code escrow should be used in the first place.

Source-code escrow can be a good protection against business-continuity risks from the licensor, but is only useful if the software that is being escrowed is of critical value for the licensee, and when they plan to use the software for a

[1] This seems quite important, as a recent analysis, conducted by Iron Mountain (a large global escrow company), showed that over 66% of deposited source-code is incomplete and over 92% of the code required extra input from programmers. http://investors.ironmountain.com/phoenix.zhtml?c=91787&p=irol-newsArticle&ID=1278015

long time. The need for escrow diminishes when the software is less critical for the licensee or when the software is generic and not heavily customized. In those cases it would generally be easier and cheaper to start looking for a new software vendor when the current vendor fails to support and maintain the current application.

4 How should a SaaS business continuity solution work?

The SaaS-model is fundamentally different from the traditional software-licensing model. With traditional software, a customer buys the right to use the software up-front. It usually includes a license to use the executable- or object-code for a pre-determined number of users, that object-code becomes in possession of the customer and is needed to run the software. After that initial investment recurring payments could occur, but those are usually only for service and maintenance, and not for the right to use the software itself. With SaaS, customers pay per-use, either on a per-user basis or on the volume of data or functions used. The big difference is that the customer does not possess the object-code on-premises but instead accesses the application on a remote server, using the internet. This remote server is managed by the SaaS-provider, either on-site or, more commonly, using a hosting-provider or even multiple external hosting-providers. Either way, the actual hardware where the software and data reside on is out of the customer's reach and control. Some SaaS-solutions even include content from third-party content providers in their SaaS-software. A customer does not have anything to do with all those external parties, and commonly they do not even know that external parties are being employed. The customer pays its SaaS-provider for access to the software, the SaaS-provider in his turn pays the different parties involved to deliver its service.

4.1 The Bankruptcy Trustee

The business continuity risk for traditional software, as discussed in section 3, is the risk of losing support and maintenance of a software package. With SaaS, the customer could face a much bigger risk; he could lose complete access to its software and data. We asked several IT-decision makers how they thought about business continuity with SaaS, and it showed that many SaaS-customers have not given much thought to this risk, and often rely on a SaaS solutions without proper business continuity guarantees. Others simply did not worry that they lose access to their application because they believe in a simple yet effective assumption, that is based on the SaaS business model itself; the SaaS model consists of a constant revenue stream. When a SaaS-Provider files for bankruptcy, a Bankruptcy Trustee (curator in Dutch) will always keep that constant stream of revenue flowing because it can be used to pay off creditors. To keep the revenue flowing, he will have to keep the SaaS application running. The Bankruptcy Trustee would cut off departments like marketing, sales and R&D, but will keep the core service itself running. The danger here is that this is based on an assumption, albeit a very logical one. The danger is that a Bankruptcy Trustee is not obliged to keep the service running, and could decide to liquidate all assets instead, leaving its customers without their software. Also, a SaaS-provider can stop its services even though it did not go bankrupt. For example, the Californian-based Platform-as-a-Service provider Coghead, which provided an online hosted platform to create enterprise database applications, announced in February of 2009 that they were acquired by SAP, and stopped supporting the application within the next month[2]. Customers had one month to develop a new application on another hosted platform and migrate their data towards it. Some companies do see these risks, and see them as an ultimate downside for outsourcing their IT to an external service provider [12], and use it as an argument to stick to the traditional on-premises software. Even if SaaS has proven to provide substantial advantages over traditional software in a number of fields. These two opposing attitudes towards SaaS could be overcome using some kind of business continuity solution.

4.2 The customer's perspective

The customer's main goal with a business continuity arrangement is of course the assurance that he continues to have access to his SaaS application and data, even if the SaaS-provider disappears. To make that work several key elements should be covered. The most important element is the customer's data. Even if access to the application has been suspended, with a recent backup a customer at least does not have to worry about losing its data, and it can start migrating alternative solution, and only lose access to his application for a couple of days or weeks, depending on the size of the data and type of application. Of course it would still be a major problem for any organization, but without access to (a recent backup of) the data, problems would be much worse; imagine a company losing all its data that resided in their CRM system. The company would not be able to service their current customers or process new leads. This could be disastrous for a lot of companies. As one of the survey respondents noted: ”When collecting and mining CRM information, continuity is essential. I guard my CRM database, why should I gamble loosing it when working in the cloud. That doesn't make sense.” Figure 1 describes a simplified version of a standard SaaS-product.

So the first step towards better business continuity would be to receive, or at least be able to acquire, regular backups of the data, preferably in a common form like XML or CSV, as Fig. 2 shows. For some SaaS-customers, this would be enough to satisfy their Business Continuity concerns, because they believe in the previously described assumed intentions of the Bankruptcy Trustee, or if they do not depend on the SaaS application that much and could easily function without it for a couple of days or even weeks.The next step towards a more complete business continuity agreement would be an agreement with the hosting-provider,

[2] See http://www.computerweekly.com/Articles/2009/02/20/234935/coghead-customers-left-high-and-dry-despite-sap-acquisition.htm

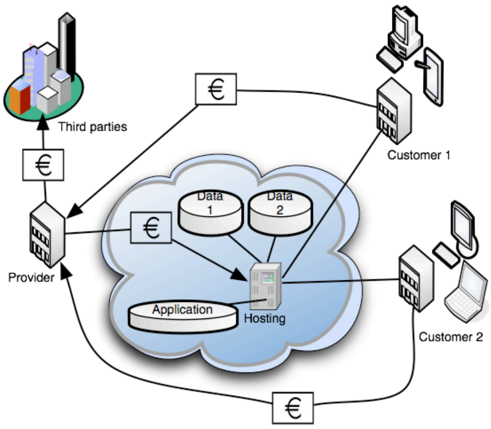

Fig. 1. A simplified SaaS-setup. Customers pay the SaaS-provider. The SaaS-provider pays the hosting-provider and third-parties. Customers access the data and application through the internet. If the SaaS-provider disappears, the hosting-provider stops hosting the application and data and the customer does not have access to either one.

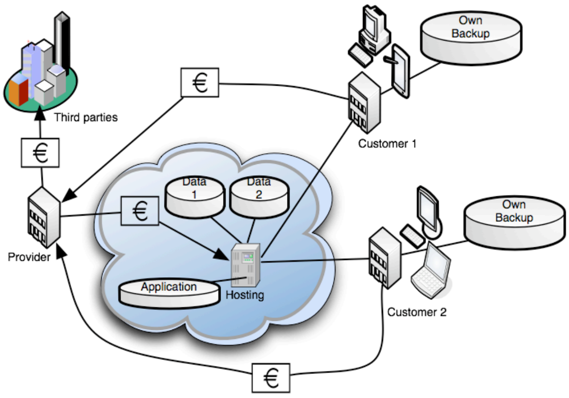

Fig. 2. A diagram showing the first step towards better business continuity, a backup of data residing with the customer. This way, whatever happens to the SaaS-provider, the customer will always have his data.

in such a way that they ensure they will continue hosting the application even when the SaaS-provider gets into financial difficulties. Such an agreement could be arranged by a SaaS-customer itself, but that would not work if there are more customers hosted on the same server, which is often the case with SaaS. Therefore, it would be a logical step to arrange this hosting continuity agreement with a separate legal entity. This can either be a commercial escrow-agent, or a foundation/fund founded by the customer(s) or SaaS-provider themselves. This separate entity can also provide some additional services next to simply continuing hosting (and providing the funds to do so). They can for example offer support for the application when the SaaS-provider fails to do so. This hosting continuity can be seen as a kind of insurance, and will be cheaper if arranged with multiple SaaS-providers at the same time, because the chance that all of the SaaS-providers fail at the same time is lower than the chance that one of them fails. Some SaaS-providers also use third party content or services in their applications. Sometimes this content is free, but frequently the SaaS-provider pays the content provider for the content. For better continuity these third party providers also have to be included in the business continuity arrangement. The last step for complete business continuity is support and maintenance.

To summarize, a recent data-backup, or the ability to acquire one on request, is the most basic and one of the most important steps in business continuity for SaaS-providers. To cover oneself completely against business continuity risks, one has to arrange continuity agreements with all third parties that the SaaS-provider uses to deliver its service. How long this continuity agreement should continue depends on the type of application, but usually a period of several months should be enough to give the customers time to migrate their data and adapt their processes to a different solution.

A different approach to virtually eliminate the business continuity risk is to provide the option for the customer to buy the SaaS-application as a standalone package. The SaaS-provider could provide a complete 'box' which includes the hardware and software to run the application on-premises. Together with a regular backup of the data, this could perfectly solve the business continuity risk. The downside with this solution is that it negates one of the biggest advantages of SaaS: the upfront capital investment. Other downsides include the fact that to successfully run and support the on-premise solution substantial knowledge and expertise about the application is required, which means the company has to hire people that possess this knowledge and expertise (or train them). Also the SaaS-vendor looses sole control of his SaaS application, which can cause problems in terms of support and updates. In most cases this would not be an ideal solution, but for specific scenarios this could be the best way to arrange business continuity with SaaS.

4.3 The providers perspective

The provider actually has no incentive of its own to arrange business continuity guarantees. The owner of a company is, from a business perspective, not interested in what happens to his customers after he filed for bankruptcy. The only

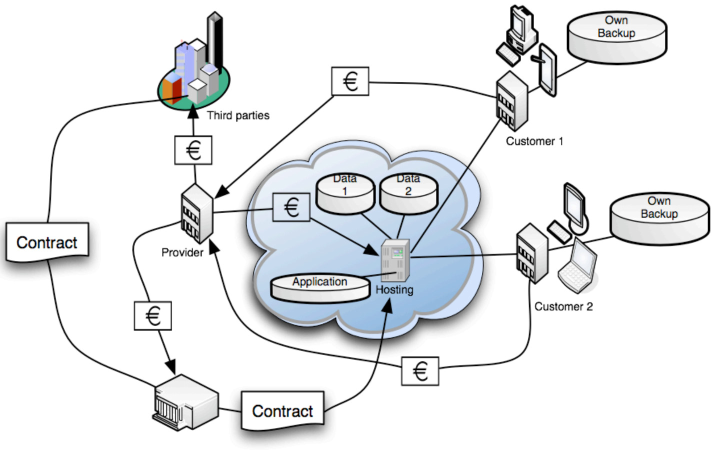

Fig. 3. The second step towards a more complete business continuity solution. A separate legal entity with an arrangement with the hosting provider and third-parties.

real incentive to arrange such a SaaS continuity guarantee is when he can use it as a selling point for his product to acquire new customers, but it only becomes a selling point if customers really start caring and asking about it. In a Business Continuity arrangement, a SaaS-Provider wants to be protected against the risk that his source-code becomes public, because that source code is one of their most important assets. Also they do not want to give other (untrusted) parties access to their servers unless it is absolutely necessary, because of the obvious risks involved. Therefore a business continuity solution should contain a trusted third party, and from that point of view most SaaS-providers would prefer a foundation that is under their own management.

Most of the interviewed SaaS-Providers explained that they have thought about business continuity solutions, but in the end did not go through with it, simply because almost none of their customers asked for it. Some customers did demand an escrow solution in their initial RFP, but never asked about it in further negotiations. This shows that a lot of customer do not worry about the financial stability of their SaaS-provider, or that they do not understand the risks at stake when migrating their IT to a cloud-based environment. Another argument that some companies used to explain the absence of a business continuity arrangement was that it would actually make them look less trustworthy, when they are the only provider worrying about its financial stability and business continuity. Of course this argument can be explained in different ways, and could also be used as a positive argument, explaining they are the most trustworthy SaaS-provider because they are the only one with a business continuity solution. Another SaaS-Provider did use an escrow-solution, but only with one of their customers because they were the only one demanding such a solution. So clearly there is not much demand for SaaS business continuity from the customers. Why this is can probably be explained using the following fact; a SaaS-Provider, like any other subscription-based service, has clear vision and control over its finances. A SaaS-provider can predict with a high amount of certainty what his revenue will be over a certain period of time. When the SaaS-Provider uses one-year contracts, then he knows how much revenue he will make for the next 12 months at any given moment. So the only way that he could get into trouble is if his costs will go up unexpectedly, which is rare because most of his costs are also on a subscription-basis. Added to that is the aforementioned belief that the Bankruptcy Trustee who, in case of bankruptcy, will keep the software running for as long as possible because it is a revenue producing part of the company. So the clear and stable financial situation of most SaaS-Providers might explain the absence of business continuity worries among SaaS-customers.

As long as there is low demand for SaaS business continuity guarantees from the customers, there will not be a business case that justifies a complete and expensive business continuity arrangement. And when there is demand, but from only one (or a few) customer(s), then the solution should be as low cost and simple as possible to be justified from a providers point of view, because only a few customers are willing to pay for it.

5 Solutions currently offered

Even though SaaS business continuity guarantees are not very common, several companies are already offering solutions. In this section these solutions are discussed, and compared with the requirements we identified for a successful business continuity guarantee.

5.1 SaaS-escrow

The most common solution for SaaS business continuity is offered by existing escrow-agents. Most of them have added a ‘SaaS-escrow’ service to their product portfolio. SaaS-escrow usually is a modified version of their regular source-code escrow. The modification often consists of the addition of a data back-up with the deposit of source-code. More complete escrow solutions also provide ‘continuation of hosting’, where they arrange an agreement with the hosting provider, that whenever the SaaS-provider gets into problems, the escrow-agent takes over the financial obligation towards the hosting-provider. The hosting provider in return promises that they will continue hosting the SaaS-application and data under any circumstance. Escrow-agents differentiate their solution by offering different extra services for SaaS-escrow, like delivering support and maintenance of the escrowed application when the escrow is released. The escrow-agent takes over support and maintenance for a predetermined amount of time. During this time, customers have the ability to migrate their data to another more permanent solution. SaaS-escrow solutions can be arranged on two different levels. The first one is a three-party arrangement with the SaaS-provider, the SaaS-customer and the escrow agent. In this arrangement the individual customer is the only customer who will be able to access the application when the escrow is released. But when multiple customers demand an escrow-arrangement, the second arrangement makes more sense; a two party ‘master-contract’ arrangement between the SaaS-provider and the escrow-agent. In this arrangement there is no limit on how many customers become a beneficiary of the escrow-arrangement, it only depends on each individual customer if they want to sign up for it (and pay the price of course). Such a master-contract arrangement is initially more expensive than a single three-party arrangement, but will make it much easier to add new customers to the arrangement, and spread the costs over all the participating customers.

With SaaS-escrow, the initial purpose of storing the source-code and releasing it to the customer on certain release-events becomes less important. Most SaaS-customers would not have any use for the source-code, because they probably do not have the hardware and infrastructure to deploy the software application on-premises. The most important function of a SaaS-escrow arrangement is the hosting-continuity agreement, because that is the only part of the arrangement that provides true continuity in such a way that the customer will not notice (or at least would not have to notice) any downtime when their SaaS-provider disappears. As an added value, the escrow company can offer support and maintenance for the SaaS application, by storing documentation and remaining in contact with key-persons involved with the software-maintenance at the SaaS-provider. So with SaaS-escrow, the escrow-agency acts more like an insurance company for hosting costs than a storage facility for sensitive information. This also creates a possible risk for the business continuity of the escrow-company itself. If the SaaS-provider grows in size, the cost for hosting the application grows accordingly. That way, it could become too expensive for the escrow-company to take over hosting-costs if a big SaaS-provider goes bankrupt.

Another possible problem with SaaS-escrow arrangements is that the solution is general and standardized, so for some specific SaaS-solutions the escrow-solution simply would not work or only cover a part of the business continuity problems. For example, typical escrow solutions do not offer support for SaaS-applications, which use a lot of third-party content in their application, because they only cover continuation of payment towards the hosting provider, but not the payment towards third-party content providers. Or imagine a SaaS-Provider who uses a lot of different hosting-providers to host their application, for example to provider better reliability and speed for customers around the world. The escrow-company then should sign a contract with every single one of those hosting-providers to be able to continue hosting the application for every customer.

5.2 SaaS Guarantee Fund

Another possible solution for SaaS business continuity is based on the idea of the so-called ’Travel Guarantee Fund’, which exists in several countries. Such a Travel Guarantee Fund covers the risk for travelers who booked a trip with a travel-agency or tour-operator who goes bankrupt before or during the actual trip. The Fund provides customers of a participating travel-agency with (financial) protection against such risks, so that customers are guaranteed that their trip is paid-for even though the travel-agency defaults. This Traveler Guarantee Fund serves as a business continuity guarantee for travel-agencies, since they protect (potential) customers for business continuity risks of the travel-agency. An adaption of such a fund could function as a business-continuity guarantee solution for SaaS-providers. SaaS-providers have a clear image of their financial situation over the coming months. They know how much it will cost to pay every third-party involved in running the SaaS application for upcoming months. With this in mind, they could set up a fund with a budget large enough to cover those costs for several months. This fund then arranges an agreement with all those third-parties, to continue their services towards the SaaS-provider under any circumstance. Since the fund is a different legal entity than the provider itself, it is not affected by financial problems or bankruptcy of the provider. In case of bankruptcy or severe financial problems, the fund can take over the payment towards all third-parties for a few months, during which customers have time to migrate their data towards another solution, or during which the SaaS-provider can make a new start again."Storing the code and data at a third party could be dangerous regarding theft of IP and setting up a guarantee-fund isn't that hard to achieve and has the advantage of (expected) lower costs." is what one of the survey respondents answered when asked which arrangement he thinks works best.

Such a guarantee fund could be set up to be very small, and only support one single SaaS-provider, so it can be perfectly tailored to support the SaaS-provider (and its customers of course) on all (business continuity) aspects. To lower the costs and efforts, several SaaS-providers could set-up a fund together, lowering the required cash-deposit per provider because it is very unlikely that the fund has to cover for all the participating providers at the same time. But a fund for multiple providers also has its disadvantages, for example: it will be less customizable towards every specific SaaS-solution. Another disadvantage is that when one of the participating SaaS-providers fail, then the others feel that they pay for its failure.

A fund has the advantage that it probably costs less than an full-fledged escrow-solution, because of its non-profit origin and low overhead costs. Also its customizability means it is a better solution for the more exotic SaaS-solutions.

At this moment, there are no known SaaS-guarantee funds with multiple participators. The problem probably lies in the initial start-up of such a fund. Who takes the initiative and invests the initial time and money in it? SaaS-providers are commercial companies, and they justify their investments and projects with a business case which predicts a profitable outcome. A SaaS-fund for multiple providers does not have any apparent extra benefits for the SaaS-provider who initiates it. Some companies have started a SaaS-fund for their own solution though.[3]

5.3 Comparison

Now we have discussed both the business-continuity options available for SaaS. Here we will compare the two and discuss their different advantages and disadvantages.

The advantages of a SaaS-escrow solution are that it is easy to set up, because it is an existing package for which providers only have to sign up once. Most escrow-companies exist for several years now and have the required legal and technical knowledge to provide a reliable business-continuity solution. Another advantage are its clear costs. Because escrow solutions are relatively standard, the costs are know in advance. The downsides are that SaaS-escrow is a standard solution with less customizability than a custom SaaS-fund, a SaaS-provider who uses a lot of different hosting parties and content providers would have a hard time finding a suitable escrow-solution. Also escrow-solutions are more expensive than SaaS-guarantee-funds because of overhead costs and the for-profit nature of escrow-companies. Survey respondents who preferred the escrow-arrangement used arguments as: "The SaaS Escrow guarantees that the source code and date are stored and will be given to the customer when things go wrong, whereas a Guarantee Fund will only give help to customers (missing the actual 'guarantee' as given by the other arrangement)", "They [The escrow company] have the legal expertise available" and "Because it is easier to setup and can be arranged on forehand".

The advantages of a SaaS-guarantee-fund are that it can be set-up for a single SaaS-solution, so that the provider remains completely in control of who has access to its source-code and other intellectual property, while still providing a workable business-continuity guarantee. Also because of its non-profit nature and very low overhead costs, all of its income will be used to serve its core activity; provide continuity, instead of overhead costs like marketing and management. Arguments favoring the SaaS-guarantee-fund were: "Without the capital knowledge of the technical staff, the code/data is of very little use.", "Storing the code and data at a third party could be dangerous regarding theft of IP and setting up a guarantee-fund isn't that hard to achieve and has the advantage of (expected) lower costs." and "It's the only option that can cover the complete infrastructure, including all third parties and resellers." Table 2 summarizes the possible advantages and disadvantages.

When we compare both solution to our findings on how a good business continuation solution should work, we can conclude that both solutions can cover the basics: they both offer data backups and continuation of hosting. SaaS-escrow does not offer extra continuation agreements for other third-parties like content

[3] To create a successful SaaS-guarantee-fund for multiple providers, the best option would be to make it a part of an existing professional association like the dutch ICT association ICT~Office.

	SaaS-escrow	**Guarantee Fund**
Advantages	Easy to arrange Legal knowledge Clear costs Experience	Complete control Customizable for specific solution (Expected) lower costs
Disadvantages	Expensive External party	Requires more effort from provider Responsibility stays with the provider No prior experience

Table 2. A table showing the advantages and disadvantages of the different solutions.

providers or extra hosting-partners. The SaaS-fund could contain all of those option, but in the end that depends on how the fund is implemented.

5.4 Necessity of SaaS continuity arrangements

As we have shown, there are several ways to ensure business continuity with SaaS. The only question that remains is: is it necessary? This is a question that has to be answered by each (potential) SaaS-customer individually. The necessity of the original source-code escrow, as discussed in section 3, is still doubted as of this day [7, 10, 11], because very few escrows are actually released. So it is only logical to doubt the necessity of SaaS business continuity solutions as well. Of course, the model of SaaS versus that of shrink-wrapped software is very different, and so are the risks at stake. With SaaS, if things go wrong, they could have disastrous consequences for some customers, because they could lose access to their software as well as their data. But the chances of a SaaS-provider going bankrupt and leaving its customers without any access to their application or data from one day to another is in fact very small, when for example considering the interests of a bankruptcy trustee. ”A great deal depends on the application itself – how critical is the data in in the application? What are the work arounds I can use to back up my information without incurring a lot more work?” one respondent answered on how he thinks about business continuity arrangements. We formulated several questions a potential SaaS-customer should ask itself to determine if it needs a business continuity arrangement:

1. How important is the software for your critical business processes and how disastrous would the consequences of losing access to your SaaS solution be? If the consequences are not that big, a SaaS continuity arrangement probably is a waste of time and money, since the chance that the solution will ever be put to use is very small. If the consequences of losing access to your SaaS-application would be disastrous, then first of all you should rethink if SaaS is the right model for your IT needs. If the benefits of SaaS outweigh the apparent risks, then one of the continuity arrangements could be a good investment.
2. How long do you expect to use the SaaS-solution?

One of the benefits of SaaS is that it is easy to switch to another solution,because of the absence of upfront investments. So if you expect to switch to a new or better solution within a year or two, then a continuity arrangement would not make sense.

3. How much would a continuity arrangement cost in relation to the cost of the SaaS-solution itself?
   The problem with a good continuity solution, is that it bears substantial costs. One of the benefits of a lot of SaaS-solutions are its low costs. So when a customer pays €100 a month for a SaaS solution, it would not make sense to pay for a continuity solution which would, at minimum, cost almost the same. Of course, the answer to the first question is more important than the costs.

These questions can help to decide if a SaaS continuity guarantee is necessary. There is one aspect of a continuity solution that we think is necessary in every SaaS-solution. That is the ability to export the customers data residing in the application at any time, preferably automatically. This export-ability is relatively easy to implement and makes the consequences of losing access to the application much less disastrous, because customers at least have their data on-site. For example, it will still be troublesome for sales executive to lose access to his CRM application, but at least he can look up information about his clients in a spreadsheet application as a basic temporary solution.

6 Survey Results

We asked several IT decision makers and SaaS providers about their thoughts on business continuity with SaaS. We asked them if their SaaS-provider provided a solution for business continuity, how important they think it is, and which of the previous alternatives they think is best. The survey results are based on 20 respondents. 50% of the respondents provide a SaaS-solution, the other 50% are SaaS-users.

The survey started by asking respondents if their current SaaS-solution (the one they currently use or provide) offered any kind of business continuity solutions. Only three of the in total 20 respondents currently offer a complete business continuity solution like SaaS-escrow or a guarantee fund. The other 17 respondents either simply answered “no” or stated that they were still looking into it. Five respondents stated that they offer data-export abilities as a continuity solution. These results are quite surprising when we look at the results of the second question which we asked to the group of (potential) SaaS-users: “Would you consider a SaaS-solution if it does not provide a clear solution for business continuity?” Seven out of ten respondents answered that they “could not take such a risk”, while the remaining three respondents answered that it depended on the specific company and SaaS-solution.

To further measure how important people think business continuity solutions for SaaS-products are, we asked respondents to rate several aspects of SaaS-products on a scale of importance ranging from “not important” to “extremely important”. The different aspects were:

1. Size of the company
2. Location of the company
3. Price of the SaaS-solution
4. Financial situation of the SaaS provider
5. Technical continuity solutions
6. Business continuity solutions
7. Data export abilities

Because of the nature of SaaS as an off-site solution, we assumed that the location of the company would be rated less important than things like price and financial stability of the SaaS-provider. We also expected that the availability of technical continuity solutions would be rated as very important. The combined results are visualized in Fig. 4. As expected, the location of the company is perceived less important than aspects like price, financial situation and technical continuity. Respondents rated data export abilities as the most important aspect when selecting a SaaS-solution. This is in line with our findings in the previous chapter, but it is surprising to see that even though it is perceived as extremely important, not every SaaS-solution offers this possibility. The next most important aspect is technical continuity. This result was expected because technical continuity solutions are solutions to prevent the negative effects of problems like power outages, loss of internet connectivity or natural hazards. These problems are a much bigger risk than financial problems or bankruptcy, simply because they are more common. The respondents rated Business continuity solutions as the third most important aspect of a SaaS-solution, with almost the same score as price and financial stability. This is quite surprising because business continuity solutions are not very common in SaaS-solutions, based on the survey results which showed that only 3 out of 20 SaaS-solutions offer some kind of business continuity solution. So there is a big difference between how important people think business continuity solutions are and the number of SaaS-providers actually using a business continuity solution. There was no significant difference between the ratings of SaaS-providers and customers.

Finally, we presented a short description of the two business continuity solutions discussed in this paper; the SaaS escrow solution and the SaaS guarantee fund, and we asked respondents to choose the one they preferred and why. Exactly 50% of the respondents preferred the escrow-solution, while the other 50% preferred the guarantee-fund alternative. Some arguments in favor of the escrow-solution were: “Because it’s easier to set-up and the arrangement is active immediately” and “they have the legal expertise available”. Arguments in favor of a SaaS guarantee fund were: “Without the capital knowledge of the technical staff, the code/data is of very little use” and “Storing the code and data at a third party could be dangerous regarding theft of IP and setting up a guarantee-fund isn’t that hard to achieve and has the advantage of (expected) lower costs”. Several respondents explained that they prefer an escrow arrangement for standard SaaS-solutions, but that they would prefer a custom guarantee fund for

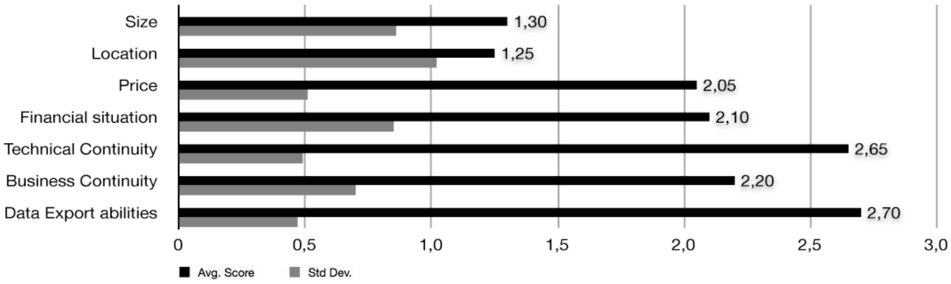

Fig. 4. A graph showing the average score of several aspects related to SaaS-solutions on a scale of importance, along with the standard deviation.

more complicated solutions with a lot of third parties, because a guarantee fund has the ability to "cover the whole chain".

To summarize, the survey showed that people think business continuity solutions are important for SaaS-providers, but currently not many SaaS-providers actually use a business continuity solution. The survey showed that both of the discussed solutions are viable options for business continuity guarantees, with equal votes, but each with their own advantages and disadvantages.

7 Conclusion

To conclude, SaaS continuity arrangements, just as every other type of continuity arrangement, is only necessary if a customer is highly dependent of the constant availability of the service that it is using. A company going bankrupt will always be a risk for every customer, no matter what kind of business it is in. To make the consequences for a customer less disastrous when a SaaS-provider goes bankrupt or gets into financial trouble, a business continuity arrangement could work, but it is very hard to provide complete protection in every scenario. In most cases, customers of SaaS-providers can find comfort in the fact that because of the SaaS business model, keeping the service running has the highest priority even if the company goes bankrupt, because it provides a continuous revenue stream. In any case, the customer at least has to make sure that he has access to his data and be able to download it. A customer should ask himself how problems with the SaaS provider would affect his own business. If a customer would get into serious problems with its own business continuity if the SaaS provider fails, then a business continuity arrangement makes sense.

As our survey showed, most providers and customers think business continuity arrangements are important, but not many providers are currently offering a business continuity solution. This probably has to do with the fact that SaaS is a relatively new phenomenon and that there is no 'best practice' yet. With this paper we hope to give some clarification on this subject, and help clarify the different options available to arrange business continuity. The types of business continuity solutions we discussed both have their pros and cons, and there is no one *best* solution. Because the SaaS model, in its current form, is relatively new, there are no practical examples to assess the effectiveness of both solutions in real life. There are no known cases where one of the two continuity arrangements were ever actually put into effect yet.

Theoretically both the escrow-solution and the fund-solution should work in a way that they provide continuing hosting for the application even though the the provider itself is bankrupt or otherwise not paying the hosting-provider. The big difference between the two is that the escrow-solution is a commercial solution, which could be more expensive because the escrow-company needs to make a profit, but offers a complete and ready to use solution with professional (legal) support. The fund-solution can be cheaper to set-up, but is more time consuming and requires a lot of effort from the SaaS-provider itself. What is the best solution depends on the type of SaaS solution and personal preferences of both the provider and its customers. We think that when business continuity solutions are needed, for most standard SaaS solutions the escrow version is preferred because of its simplicity and low effort requirements. When the SaaS solution is more exotic or needs more specific arrangements with a lot of third parties or for a difficult infrastructure, the fund-solution would be a better alternative.

The most important factor in business continuity is data access. A customers data is also stored on the servers of the SaaS-providers. Usually the provider will take good care of back-ups and everything so that they will never loose their customers data, but those backups often still reside with SaaS provider and not the customer himself. Therefore we think, and our survey showed, one of the most important aspects of business continuity is a regular backup of the data towards the customer himself, so that they always have a recent copy no matter what happens to the SaaS provider. This data backup option should always be offered by every SaaS provider, even if there is no official business continuity arrangement.

References

1. N. Abdat, M. Spruit, and M. Bos. *Digital Product Management, Technology and Practice: Interdisciplinary Perspectives*, chapter Software as a Service and the Pricing Strategy for Vendors. Advances in E-Business Research (AEBR) Book Series. IGI Global. In Press.
2. A.N. Hiles. Service level agreements. *The TQM Magazine*, 6(2):14–16, 1994.
3. A. Dubey and D. Wagle. Delivering software as a service. *The McKinsey Quarterly*, 2007.
4. J.M. Kaplan. Saas: friend or foe? *Business Communications Review*, 37(6), 2007.
5. W.L. Currie and P. Seltsikas. Exploring the supply-side of it outsourcing: evaluating the emerging role of application service providers. *European Journal of Information Systems*, 10(3):123–134, 2001.
6. M. Chen, A.N.K. Chen, and B.B.M. Shao. The implications and impacts of web services to electronic commerce research and practices. *J. Electron. Commerce Res.*, 4(4):128–139, 2003.
7. J.L. Mezrich. Source code escrow: An exercise in futility. *Marquette Intellectual Property Law Review*, 5, 2001.

8. J.E. Raymond. Software licenses, source code escrows, and trustee powers under 11 usc sec. 365. *J. Business Entrepreneurship & Law*, 1, 2007.
9. P.A. Pappous. Software Escrow: The Court Favorite and Bankruptcy Law, The. *Santa Clara Computer & High Tech. Law Journal*, 1, 1985.
10. W.D. Denson. Source code escrow: A worthwhile or worthless investment, the. *Rutgers Bankruptcy Law Journal*, 1, 2002.
11. S. Helms and A. Cheng. Source code escrow: Are you just following the herd?, March 2008.
12. A.H. Spiotto and J.E. Spiotto. Ultimate downside of outsourcing: Bankruptcy of the service provider, the. *American Bankruptcy Institute Law Review*, 11, 2003.