Using E-markets for Globally Distributed Work

Jos van Hillegersberg and Chintan Amrit(✉)

IEBIS, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands
{j.vanHillegersberg,c.amrit}@utwente.nl

Abstract. For over a decade, dedicated E-markets have been facilitating globally distributed systems development by enhancing the traditionally high-risk global sourcing processes. At the same time, the success and potential of E-markets for sourcing project globally can be questioned, as E-markets embody a variety of temporal, geographical and socio-cultural gaps. To study the effectiveness of the mechanisms offered by the E-markets, we ran a field experiment in which four development teams worked for 10 weeks to have a software development product designed, programmed and tested by remote developer(s) using an E-market. Three out of the four teams managed to deliver a successful product within time and budget. This result exceeded our expectations and contradicts the critical observations and opinions in several blogs and news articles. We find that for effective e-Market sourcing a skilled customer team with competences including vendor selection, software contracting, software requirements specification, development methods, cross-cultural and virtual communications, use of various cloud based tools, frequent functional and non functional testing are necessary.

Keywords: Global outsourcing · E-markets · Outsourcing risks

1 Introduction

The risks associated with globally distributed development of software have been extensively studied. Risks specific to the globally distributed setting are often classified based on the temporal, geographical and social cultural dimensions and their potential impact on Communication, Coordination and Control [1, 2]. Examples of Communication risks include lack of synchronous communications (temporal), limited face-to-face meeting time (geographical) and cultural diversity (socio-cultural) [1]. Coordination risks include lack of overlapping work hours (temporal), unclear task responsibilities (geographical) and insufficient knowledge and information sharing (socio-cultural) [1, 3]. Finally, Control risks include delayed feedback (temporal), challenges in maintaining stakeholders' commitment (geographical) and lack of equal domain knowledge (socio-cultural) [1]. Traditionally, mitigation strategies have focused on decreasing the gaps by carefully selecting vendors in lengthy bidding processes to assess socio-cultural fit and appropriate domain knowledge. During development, project partners have attempted to lower risks by investing in building relationships and trust between dispersed team members. Measures include frequent site visits, cross cultural training, social events, and placing individual developers for longer periods at the remote site.

I. Oshri et al. (Eds.): Global Sourcing 2015, LNBIP 236, pp. 206–227, 2015.
DOI: 10.1007/978-3-319-26739-5_12

While sometimes effective, these practices take considerable time and significantly increase costs. In extreme cases, mitigation costs may completely ruin the potential cost and efficiency gains of distributed development. Furthermore, these practices are mainly fit for larger projects where supplier and vendor will work intensively together for longer periods of time. There is clearly a need for more efficient and effective mechanisms for small to medium sized projects with short-term duration.

For several years, dedicated E-markets have been targeting at facilitating globally distributed systems development. E-markets support key sourcing processes such as vendor search, contractor and contractee ratings, matching, bidding, communicating about requirements, defining milestones and deliverables, conflict and dispute resolution, escrow services and settling payments. Through these services, E-markets claim to greatly enhance the traditionally high-risk global sourcing processes. Dedicated E-markets for sourcing distributed work have been around for years and have seen their business growing. Well-known examples of E-markets for sourcing of online work are Elance, Odesk, Guru and Freelancer. Elance was founded in 1999, Odesk in 2005. The two companies merged early 2014 into Elance-Odesk, generally seen as the global leader in online work. Elance-Odesk facilitates a community of 9 million freelancers and 3.7 million businesses. Elance-Odesk claims 2.7 million annualized job postings with workers making $900 million in annualized earnings [4].

At the same time, the success and potential of E-markets for sourcing project globally can be questioned. In 2013, Elance-Odesk holds less than 1 % of the overall global staffing market ("Why Freelancers Are Dismayed At Elance's Merger With oDesk," n.d.). Their share of the market is only a fraction of the total software development business. A simple calculation demonstrates that on average the 9 million freelancers make only 100$ annually. It is very likely many of the accounts are dormant or hardly get any business while bidding fruitlessly on posted projects.

In recent blog post and magazine articles, the usefulness of E-markets to run high-quality software development projects has been questioned. For example, a blog post on http://blog.launchastartup.com/ lists several drawbacks in the current practices of using E-Markets. First companies do no spent time to properly describe jobs and post ill-defined projects. In an attempt to bid fast, and spent very little time, vendors offer low quality bids. Quality bids may get lost in the high volume of bids sometimes even automatically generated. Price tends to dominate and results in a non-ideal match between contractee and the contractor. Both parties muddle through towards a solution, unwilling to go into the time intensive bidding process again, but at the same time remain unhappy with the current progress. Developers from low cost countries, working for as little as 2 to 3 dollars an hour, scare away freelancers with higher hourly rates. Outsourcers complain that it is very hard to find both high quality and reliable developers. They question how the high reputations scores have been built up. The low chance of winning a project may ultimately cause the more serious vendors to lose their interest. George Anders in a recent Forbes article writes: "These hubs for freelancers rely too much on the joyless mechanics of market matching … and not enough on the quieter, friendlier dynamic of developing long-term trust between select employers and specialists who get to know one other" [5].

Thus, currently it is highly debated whether the mechanisms E-markets offer to bridge temporal, geographical and socio-cultural gaps that lead to successful system development projects and bring value to both outsourcers and vendors. As our literature review will show, researchers have not adequately addressed these questions.

In this paper we study the potential of E-markets to mitigate the common risks faced by globally distributed software development teams working on complex and domain specific requirements under high time and budget pressure. To study the effectiveness of the mechanisms offered by the E-markets, we ran a field experiment in which four development teams worked for 10 weeks to have a software development product designed, programmed and tested by remote developer(s) using an E-market. While each team worked on exactly the same initial project definition, budget, deadline and scope, they choose to make use of different leading E-markets, like Freelancer, Guru and Odesk. The teams were also given the freedom in how they used the E-market support for finding an appropriate vendor, communicating requirements and controlling progress. During the project, the teams kept a log of their experiences in tackling the typical risks in global sourcing. Three out of the four teams managed to deliver a successful product within time and budget. In this paper we discuss how various E-market features were used and evaluated by the development teams. We review the strategies used to select a trusted vendor and build a fruitful relationship, and reflect on the current features and differences among the three E-markets that were used. We then discuss how E-markets could enhance their support for distributed software development even further to facilitate ever larger and more complex projects, while successfully mitigating the risks of temporal, geographical and socio-cultural distance.

2 Related Literature

Software development was a popular activity already with the first emerging E-Markets for global work and services in the early 2000s. : shows the key features an E-Market facilitating the customer and service provider relationship [6]. The service provider builds up its profile consisting of his own company information and skills and project evaluations that are built up over time, provided by earlier customers and verified by the Electronic Services Market (ESM). As the E-Market also handles payments, the business results (such as annual reports) can also be generated by the E-Market and consulted by the customer. The contract is the result of a fully open or by-invitation bid process and kept by E-market specifying price, duration and important milestones and payment terms. Not shown in Fig. 1, but available in several E-markets is the feature to rate customers by providers. With this feature providers have information available to evaluate if they want to work for a customer.

Aguinis and Lawal discuss the “ELancing” phenomena, the various features of E-Markets and its implications for research [7]. Based on the E-market features, they argue that none of these are present in traditional outsourcing, offshoring, temporary work, teleworking and independent contracting. Thus, sourcing work through E-Markets is truly different from other modes of sourcing work and has its own set of practical challenges and research questions. “The marketplace plays a key role in

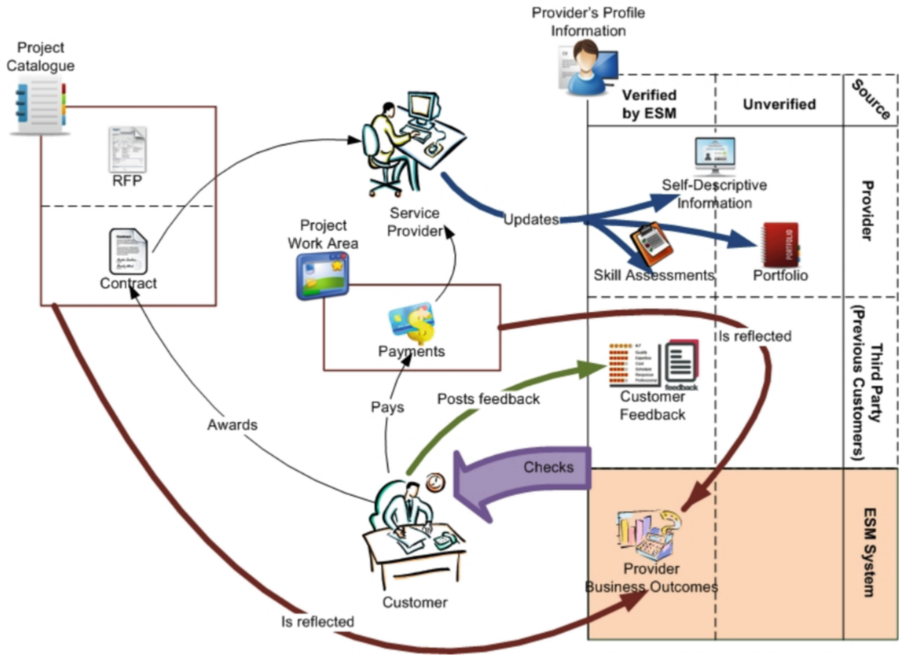

Fig. 1. E-Market services facilitating Customer and Service provider interaction (source [6])

matching employer with employee, assigning work, reviewing performance, and allocating rewards. This is why the marketplace keeps a portion of the profits" [7]. E-markets compete in various ways, e.g. by varying their commission rates, typically between 5 – 15 % of the total transaction value to be paid by the contractor. E-markets have started to also offer hourly rate projects in addition to the fixed price model. To address the large volume and fierce competition in open auctions, E-markets have started to offer private auctions by invitation. E-markets also vary in the richness of the profile information they provide including various statistics on earnings, repeat business and customer ratings of past projects. Several E-markets offer communication tools and work monitoring tools. Furthermore, they offer ever more advanced payment schemes and customizable milestones. Escrow services are offered by most E-markets, sometimes with mediation facilities in case of disagreement.

Most studies have examined the workings E-Markets from an outside perspective; looking at the number of providers and customers, transactions, features, use of profiles, etc.

Taking a provider-centric perspective [6], develop and test a model of the potential impact of provider profiles on their business outcomes. Based on an analysis of business results of providers active on a large E-market with more than 50.000$ turnover in 2011, they find that external profile information positively impacts business results, while internal information did not impact results. External information components mainly imply the possession of technical abilities, and experience by the provider, as specified by the provider. Internal information constitutes feedback ratings, recommendations and text comments submitted by a provider's previous customers.

Gefen and Carmel (2008) examined the proposition that information technology providers from low-wage nations can now underbid providers from high-wage nations and win contracts [8]. They study an online programming marketplace and analysed the history of transactions [8]. They find that the strongest determinant of the winning bid still is client loyalty: the client gives very strong preference to a provider with whom there has been a previous relationship, regardless of whether the provider is offshore or domestic.

Walter (2013) explores the importance of E-markets characteristics and software development project success factors on project success in his thesis on "Success Factors in Leveraging Freelance Marketplaces in Software Development Projects" [9]. He analyses the literature and dozens of blog posts on the use of E-markets for sourcing software development, both from the buyer and contractor perspective. Based on this data Walter develops an E-market software development project success model [9] (Fig. 2). The model is tested using a survey among users of E-markets. The initial results point at Cost control factors, Project efficiency factors, Qualifications of the buyer and Focus on product quality, as the important project success factors. For the E-market characteristics, both Website functionalities and Terms agreed upon mattered. However, the number of respondents to the survey was small (36 respondents) and obviously the depth of analysis was limited as a result of the length of the questionnaire.

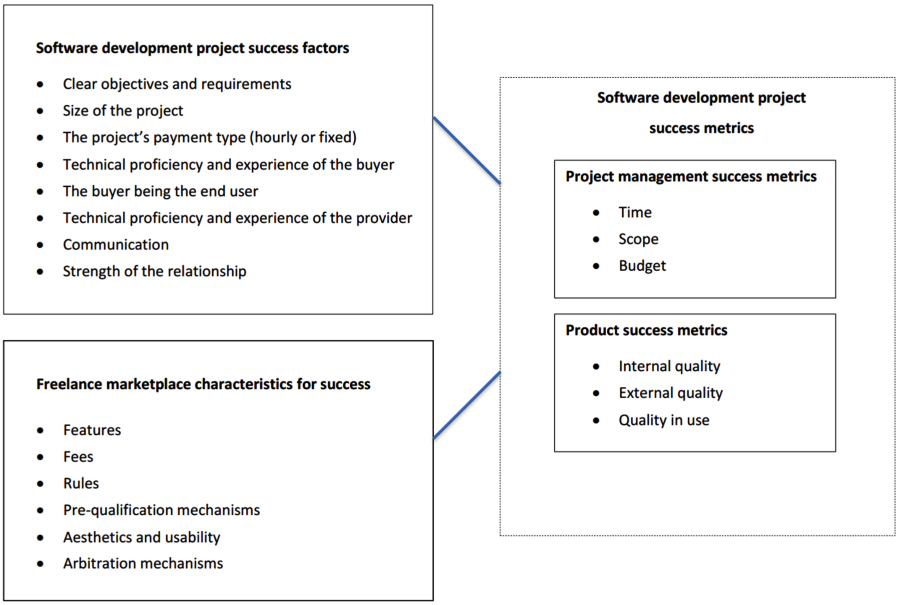

Fig. 2. A model to study project success of software development using E-markets. Source [9]

In our research we explore the relationships between E-market characteristics, software development project success factors and project success in more depth using a field experiment. The experiment and the results are described below.

3 Research Method: Field Experiment

To get more insight in project success factors and the role of E-markets in achieving sourcing success, we designed a field experiment to gain detailed experience with software development using E-markets. In the field experiment, four teams of 5 to 6 advanced MSc students enrolled in our Global Project Management course of the Business & IT MSc program were given the assignment to develop a system using a software development vendor via an E-market. Each of the teams was given the same assignment to develop a software game called Xchange. At the first course session (April 2014), the instructors played a paper-based version of the game with the group to illustrate the game and its mechanics. Next, the teams were told that they had 10 weeks development time and were allowed to spend €500 each to develop a web based version of the game using an E-market. The teams were given the freedom in what market to use, what tools to use and how to organize the development process. However, they were not allowed to spend more than the budget or to overrun the schedule. Also they were not allowed to code by themselves, as the goal was to gain experience in transferring requirements offshore online, and to effectively manage a global project using E-market functionality and other online communication and project management tools.

Each team was required to maintain a detailed log of their experiences and lessons learned. At the final course session (June 2014), the four teams presented the working game software developed and the process experiences, assessed product quality and lessons learned. The two instructors assessed the software products and project reports of the four teams and the software products that had been developed. Given the stringent parameters, it was surprising that three out of four teams managed to deliver good quality working software within time and budget, while one team delivered a partially working product. Considering our initial scepticism of sourcing using E-markets, the rich requirements and limited budget of only 500 euro and tight schedule of only 10 weeks, the results greatly exceeded our expectations. Below we highlight the experiences of the four teams based on the rich materials about their projects that they delivered. We have structured the results inspired by the research model by Walter [9] and the sourcing phases as described in [6], organizing the findings of each team in the following area's:

1. Choosing a platform and contractor
2. Contracting
3. Specifying Requirements
4. Development process
5. Project Success

We present the findings in each of this area's below.

3.1 Sourcing Field Study Team Experiences

Team 1. The experiences of team1 are illustrated by the excerpts from their logs and final report and summarized here as first person accounts:

Choosing a Platform and Contractor. We decided to use oDesk as a platform, since it was the only platform where we were able to find developers capable of working with the rapid prototyping framework MeteorJS. We received the first applications within a few minutes. We responded to serious offers and also invited some developers directly to our job description which we already found using the oDesk search engine. After filtering out some "bad apples" and excesses in price offerings, we arrived at a short-list of two candidates. We set up a Skype meeting with both to discuss in more detail our offer, inquiring about their application and asking them to provide an outline of their work during the development. In this way, we enabled them to decide on deliverables at milestones themselves, to put them in charge of their own agenda. Deciding ourselves on deadlines that are impossible for a developer to reach is useless, as the developer might be culturally restricted to refuse which eventually leads to failing deadlines and miscommunication. The final candidates (with their info-line on oDesk) were:

1. Nivov H.: Furniture Assembly Expert.Theoretical Physicist.JavaScript Developer
2. Poojan S.: Full Stack Developer: Node.js, AngularJS, Android, PHP, DB, Linux.

We did a little background research on both:

Nivov H. seems not to use his real name on oDesk, comments left by others named him Mushex or Allesandro. We did found his LinkedIn and GitHub account, where he posted some interesting stuff but not very professional, like a Web 1.0 theme for the Twitter Bootstrap Framework (http://code.divshot.com/geo-bootstrap/).

Poojan S. impressed us with a free demo. He kind of had to, because this would be his first project on oDesk. His written English vocabulary was impressive, however, during our Skype conversation he sometimes had trouble finding the right words. He did use his real name though, and has an impressive GitHub portfolio while his LinkedIn-profile looked professional with a clear picture of himself. We ultimately decided for Poojan S. as a contractee. Nivov H.'s way of implementation was not in MeteorJS, but an equivalent which was less known and he did not provide us with similar projects as reference. In our last interview before we contracted Poojan, we asked him by when would he be able to complete the assignment, to which he said he would be done within a week. We were a little sceptical about that deadline, so we asked him to keep the next weeks also available for bug fixing. Also, we supplemented our requirements to provide more or less as a contract, based on his promise.

The candidate we chose was from Nepal and had no former experience on oDesk, which could be a potential threat. Because his other Internet profiles (LinkedIn, Google Plus, Github, Facebook) seemed consistent and he made a demo for us, we chose to work with him. Getting in touch was very easy and he responded fast to our questions/remarks, except when they were too critical.

Contracting. We agreed with the developer on the following terms of payment:

- 50 % upfront
- 50 % when all of the following is true (we define the following as "complete")

1. We have received the code of the product.
2. We have verified the code of the product.

3. All functionalities and other requirements of the product as described in this specification document are implemented and working.
4. If anything above is not met, the product will be developed until all holds true before June 10th, 2014.

Specifying Requirements. Because we did not have any knowledge about what quality we could expect from the applicants, we decided to approach this outsourcing assignment in as simple a way as possible. With the limited budget, time and need for high quality code this seemed the best choice to us. We wanted to implement the basic features including some extra requirements we made using simple visuals in order to enable the programmer to spend more time on the code quality than in understanding the game itself. We provided the outsourcee with a 26-pages long requirements document.

We specified the system boundary, rules, task descriptions, mock-up and some quality requirements. This way it was clear for both parties what we wanted in the game. Our software developer also said he had a clear understanding of what had to be done. The other applicant on our outsourcing platform was also impressed by our specifications, although this can be a result of the politeness of the candidates.

Another important aspect was that it seemed like the developer did not get the exact idea behind the game. He understood simple functionalities and was able to program them, but he had more trouble with correctly implementing more business-like functionalities closely related to the game rules, like when something is a profit or a loss. Our job was to be very specific and explicit about the more complex functionalities, in order that he could just follow detailed instructions instead of implementing high-level goals.

Development Process. Before the decision to contract the developer was made, we decided to use a waterfall approach. We agreed to not use an agile approach because it seemed to be more time consuming and we thought was possibly less suited for small software projects, like our game. We already completed most of our requirements, and as we were able to complete the requirements within a week so we chose to continue with the waterfall approach. After the outsourcing project started by providing Poojan the final requirements specification document (including the formal contract requirements), our team awaited the first delivery were he said to be done with everything. This milestone was not met, and we waited for a few more days. After this extended milestone was also incomplete (there has been source code committed, but not yet functional nor testable), we reminded him of the contract requirements in terms of delivery. We initiated a Skype meeting with our team and Poojan, in order to enable him explaining the situation to us and give us the reason of his delay. He was not very talkative after our reminders, and promised us (again and again) to keep working, giving all his time, and finishing the whole project within a couple of days. After missing the second deadline, nobody believed his deadline promises anymore, but it worked to keep him developing and enable us to review his progress each couple of days. It was quite a challenge to have a conversation with him. This was because his spoken English level was quite low, and it seemed that he did not always understand what we meant, as he was just responding with a short “ok” several times. He mainly blamed slow deployment and testing on a slow Internet connection. While he became aware of his delay, we decided to report missing functionality, issues and quality considerations by using the GitHub issue tracker, which proved to be working extremely well.

After this meeting on the first of June, Poojan showed incredible performance, it appeared like he was working day and night on our project and directly solved newly reported issues really fast. We were able to constantly monitor progress and test on the live environment he provided (upon each code commit, he also deployed the current version on the Meteor live site). This allowed us to verify if issued were solved. Ultimately he delivered according our own requested delivery date and additionally implemented some quality issues we reported. Neither of us explicitly mentioned it, but we regarded this reasonable since he exceeded his own promised delivery dates several times. When looking at the requirements we did specify, we quite benefitted from the established, explicit contract requirements detailing our general game specifics and quality requirements.

Use of Tools.

- Balsamiq - For Mock-ups supplementing the Requirements Specification
- oDesk – Outsourcing platform, contract agreements & Issue solving
- Skype – Communication with applicants
- GitHub –Version Control platform, issue management
- MeteorJS – Web Framework based on JavaScript
- JSHint - Code quality verification tool
- Facebook Group - Own primary communication channel
- WhatsApp Group - Secondary communication channel
- Google Chrome Developer Console-Simple debugging when testing delivered work.

Everyone in our team was familiar with Balsamiq mock-ups, so we could easily create simple interfaces for our Requirements document for this (Fig. 3).

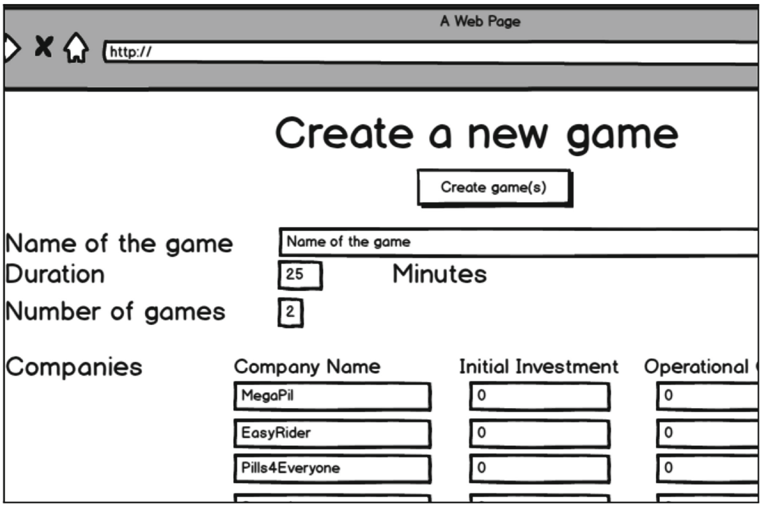

Fig. 3. Sample screen design in balasamiq (Team1)

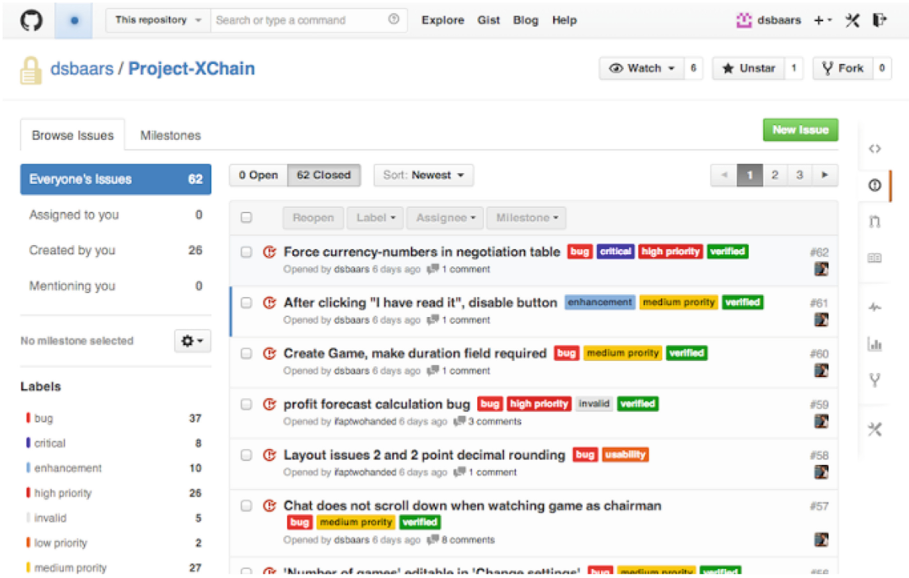

Fig. 4. Use of GitHub for issues reporting (Team1)

We used GitHub as code platform and issue tracker. This went very well, especially the simple tagging system helped mark the priority of issues and the status of verification (Fig. 4).

Sometimes we used Skype to talk to the contractor. Sometimes the sound quality was not that good, but fortunately our programmer responded on oDesk very fast when he was available. We used JSHint to detect errors and potential problems in the written JavaScript code. The final delivered code did not contain errors, but during the projects we sometimes checked specific commits, which were not always without errors. For our own documents we used Google Drive to work with multiple people in one document.

We used a Facebook Group as our main communication channel, and scheduled meeting. For urgent matters we also used a WhatsApp group.

Project Success. After some intensive 1, 5 weeks of programming in the beginning of June, the last issues were resolved on June 11, and on June 13 (after final testing) we were able to confirm the delivery of our project according to our specification. Reflecting back on the method we initially used, we switched our waterfall method after the first "function complete" deliverable to an agile way of working, using milestones as equivalent to Scrum sprints. Although the basic development method was still a waterfall, we did not come up with extra requirements or functionalities during the design (except for frequent quality issues), our communication and collaboration was quite agile due the high intensity of developing features and testing these immediately.

Team 2.

Choosing a Platform and Contractor. Ashort version of the requirements was created and uploaded on Guru.com together with a description of the game in order to find a developer. The selection of the offshore developer team was a difficult and challenging task with the limited amount of budget and strict deadline. With our advertisement on Guru.com, we got 21 quotes initially. It was a tricky task to discover the right developer/ development team with right development ability and communication skills. With the number of quotes, we also got a large set of questions to answer which resulted in an immensely time consuming task. Several developers and companies reacted to the project on Guru.com. Most companies we negotiated with required compensation far above our budget. At last, with the discussion, communication and negotiation, we finalised the negotiation with one development company called GlobusSoft from India that agreed with our limited budget and timeline. Globussoft was the only company that was willing to develop the game for us for the budget of 500 euros. Due to the time constraints we decided to go with this company. A Skype call was made with Globussoft to make sure the developer and our group had a common understanding of the game to be developed. Globussoft made a video showing the functionality of an initial version of the game. GlobusSoft is a company, which started in 2009 and is located in Bhilai, India.

We used Guru.com as a platform to attract our developing party, as we were very satisfied with the platform. Not only did it offer us a marketplace, it also offered basic project management functionality. This made the use of a tool like Basecamp redundant, so we ditched that as soon as we saw the same functionality in Guru.com. Guru.com also offered a safe payment method, though it was a bit more expensive than direct transfer, it did help raise trust from the developing party that if they would do a good job, the money would be paid. It was also possible to pay parts, like per milestone (Fig. 5). We found it very important to check the skills of the programmer before hiring him, by checking his or her previous jobs.

There were a total six people in offshore team. Two of them were developers and one was a support person. For our frequent discussion, we had mostly interacted with these two developers. Communication with support people was only done at the start and at the end of the project.

Contracting. Payment was decided to be 500 divided by the number of iterations (denoted by P). After the first iteration, if the result were not to our satisfaction, we would pay the vendor half of P. He would then have the choice to continue working on the project or to quit. If he wanted to continue and his argument is to our satisfaction, we would continue working with him. Why should we pay the developer half of P if his work is not satisfactory? A number of reasons: (a) it is only fair, the developer has spent time on the project and (b) if we would not have paid him, we would be disrespecting his work leading to a negative comment on our profile, this could potentially result in no developers willing to work for us. To verify that the developer was reliable and capable, we tested him by asking to implement a part of the game in an hour or two. We

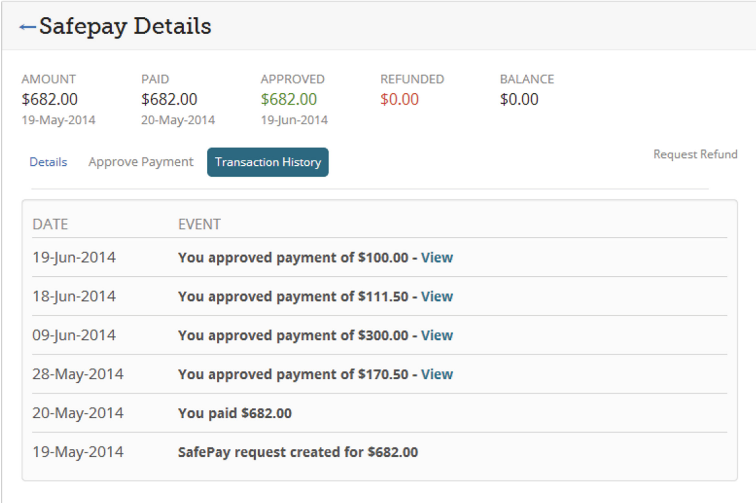

Fig. 5. Safepay payment schedule in Guru.com (Team2)

paid the dues after each milestone and kept the last 100 dollars until the moment that we thought we were satisfied with the game.

The payment plan and iteration planning did work as expected. The developing party even was consistently a week ahead of schedule. This made us happy to pay them for their services. At the beginning of the negotiation with the programmer we had some difficulty in knowing if the price for him was right. We needed to repeat our questions several times because he was never replying with a direct yes/no. To avoid misunderstandings about the price, we learned that it is better to be very clear from the beginning of the project about the maximum price we are going to pay.

When we hired GlobusSoft, we decided upon a price for fulfilling the requirements, but we forgot to make clear agreements on when payment will take place and when the code must be handed over. This resulted in the developers repeatedly asking for part of the payment and us repeatedly asking for some of the code. In the end, we received the code, but we were unable to have a look at the code before the entire project was finished and the full payment was done.

Specifying Requirements. In addition to screen mock-ups, we provided UML based activity and class diagrams to specify the design of the system. At the beginning of the project, it was difficult to communicate with the developers, as there were many questions and a lot of confusion about the financial data sheets implementation. But, with the start of the development process, the problems got mitigated. The weekly video demo from the offshore team and the feedback from our team to the offshore team had been useful to keep the developers on track and our team satisfied. We made a big requirement design, very detailed, in the beginning. Also we specified a lot of non-functional requirements in the beginning, some of them were not useful because they were too specific and, after the beginning of the programming part, we found out that they were not that usable. We learned that it is better to have fewer non-functional requirements in the beginning of the project and specify them as the project progresses.

Development Process. We devised the development process as a combination between the waterfall and the agile approach. Because of the challenges posed by the global aspect of the development of the game, such as the time difference, cultural difference and language barriers, waterfall has the advantage that specifying and designing the game beforehand thoroughly before actual implementation, the developer knows what has to be developed. This will not leave room for the developer to come up with his own ideas for the game, which could be considered a disadvantage as well as an advantage. As requirement engineers and designers are not fully aware of all technical capabilities, the agile approach enables the developer to come up with his own ideas and (possible) improvements for the game. By going completely agile, we think that the challenges posed by the global aspect of the development of the game will outweigh the advantages of agile. Agile requires frequent communication between all stakeholders of the development of the game. This can get time consuming because of the fact that much of the communication will be asynchronous.

We received a 2nd video with a demo from Globussoft to show the progress of their work. We had a feedback call with Globussoft to discuss the progress that was showed in the videos. After multiple iterations (an iteration taking roughly a week), the game was finished and installed on our server. On June 13th we received a link to play the game and on June 19th code was delivered when the last payment was made.

We were a very heterogeneous group composed by people from different nationalities, different backgrounds and therefore different ways of thinking. This was an advantage, on one hand, because we had different perspectives and different opinions about the job, but on the other hand it was a disadvantage due to the difficulty to come to an agreement in some circumstances. We improved the relationship with the programmer by replying quickly to every question that he had. We also gave feedback on the software's milestones in a complete and quick way. We found out that it is important to make a connection with the programmer (by talking about non work related topics).

Use of Tools. We let the developer decide the tools to be used, as we did not force any tooling upon the developer and let him use the tools he is familiar with. We used various media to communicate with the developer. For face-to-face communication and for feedback sessions we used Skype. We used the project management tool called Basecamp to communicate the project details. We attached documents from Google Drive to Basecamp. We used Google Drive to share documents for the project internally. These consisted of requirement documents, sketches of the interface, etc. Some of these, not all, were shared through Basecamp with the developer. We contacted them for every scheduled meeting on a weekly basis with a Skype call and privately with text messages using Skype. We reported our remarks during the call, but we also used an Issue Tracker that we shared with them on the Google drive. We always used a mix of synchronous (Skype calls) and asynchronous (e-mails) for our communication, both of them were very useful.

Project Success. It was aimed to construct a global team consisting of developers of the developing party and our team, in order to enable the developers in the team to come up with new ideas and possible improvements. This construction of a global team materialise, as there was a clear distinction between them and us. They were hired to develop, and we were their customers. In general, we found it easy to communicate with the developers and explain our concerns. However, we faced difficulties in vocal communication. Written chat was more effective in our case than the voice chat. A possible reason could be the language and diverse accents. We noticed that it was difficult for the offshore team to understand our accent and vice versa. However, from our team's perspective, this problem was somehow less severe as one member from our team was able to understand and capture the offshore teams' accent and speed of speech. After the communication and language, the major coordination issue was difference in time zones. The problem of time zone difference between our team and the offshore team was less intense because of two major reasons. Firstly, the time zone difference between India and The Netherlands is not that large and was successfully synchronized. In addition, we observed that the offshore developer maintained a very professional (boss-employee) relationship with our team where communication was quite precise.

The assessment of software quality metrics such as source code usability, reusability, maintainability, complexity and a number of other metrics of code, are important. This is especially the case since another party has created the code, it is important that the code is understandable, in order to be able add functionality and perform maintenance.

The code for the game was written in Java. The code was imported in Eclipse to be able to run some metrics on it. When importing and looking at the code, a big downside was discovered immediately. There was no documentation in the code whatsoever. For the purpose of the source code quality assessment, we used the Eclipse Metrics plugin tool that. The quality of source code was measured on a number of different general metrics e.g. number of packages, number of methods, number of interfaces, lines of code to the specific quality metrics e.g. McCabe cyclomatic complexity, weighted methods per class (WMC), lack of cohesion of methods (LCOM), etc. The final assessment of the code quality was not so good. The metrics indicated some weak points. Even though the results from the metrics were not dramatic, it was obvious that it will be hard to add functionality to this code or maintain or update it in any way. This was caused by the complete lack of documentation Javadoc in the code. The delivery of proper documentation was not put clearly in the requirements, which apparently was needed to get proper code quality.

Team 3.

Choosing a Platform and Contractor. We quickly found out that Odesk was our preferred website for the rest of the project. Most of the replies on our first Odesk posting came from Indian and Pakistan developers. After several Skype sessions we selected a programming agency, called Websumster Technologies. Initially, we thought we hired a female programmer, but after the first Skype session our contact changed to a manager from the same company. It took some time for us to clarify that we wanted our first sprint needed to be finished within one week. During this week the manager told us that they were working on the project, and every time we spoke he said ‘within two days you can see some progress’. However, after 1.5 weeks we did not see any progress at all, and started pushing them to show us something. It took two hours, but then they sent us a URL to an online environment where we could see their progress. When we took a closer look, it turned out the code was copied from a chat box tutorial at CSSTricks.com. That was reason for us to stop working with this company, because they could not give us a reasonable explanation for copying the code. We told our story to the Odesk customer service, and they have banned the initial developer’s account. After 1.5 weeks we had to start our programmer selection process all over again. This time we changed our approach, and asked developers to show us something (e.g. a html conversion of the mock-ups) before we decided to hire them. This led to one Ukrainian programmer who built almost 50 % of the total functionality of the first sprint in just a few hours, in order to convince us to hire him. After a compelling Skype conversation we chose to hire this programmer. He delivered the sprints as promised, and even surprised us by hiring a special designer to redesign the whole layout without us having to pay more. The Ukrainian developer was quite young, was experienced with web development and asked the right questions. Cultural differences were not present: just like our local team members, he was straightforward, did not need small talk and kept his promises. Another advantage was the fact that we could always access his server, so we could have an update whenever we wanted.

Contracting. Odesk was the platform of our choice to recruit and pay our programmer(s). Before we made a decision about the platform, we posted jobs on five different platforms. On almost all of the platforms, we got bids or offers, but we saw the most potential in the Odesk platform. Programmers reacted quickly and provided realistic offers. The fee Odesk charges is at the higher end, but very clear: 10 % of the amount is paid to Odesk. Programmers can add 11.11 % to their price for the final price. For us, this fee was a nice bargaining leverage. Overall, Odesk worked quite intuitively, it was easy to access and handle complaints.

Specifying Requirements. We would be lying if we said we had no problems in the area. There were a couple of really basic things we overlooked in the first meetings with our foreign programmers. The lesson we learned from this was fourfold. (1) Make sure that the home-team has reached a complete consensus on what agreements need to be made and what the negotiation scope is. Write it down and use it as a guide through the meeting. Possible elements could be price, deadlines, getting updates, availability for questions, tools, delivery of end product etc. (2) Use video meetings as a way to bond with your away-team and do not worry about getting a bit informal. These moments create trust amongst each other and create a better working environment. I.e. we used some Ukrainian friends to write texts to our programmer in Russian. (3) When agreements are not met, take action. Be respectful, but very clear about what the expectations are. Think about how you would react if the same situation would happen in your home-team and use it as a guide to overcome the challenges. Be quick to act if the challenges are not being tackled and resort to plan B. (4) Do not position yourself as dependant on the away-team. Setup your project in way that allows you to change plans and switch programmers if necessary. Using this as requirements makes it clear for both sides.

Development Process. The project was divided into five steps, and in the week of May 2nd, we posted the first step on several websites. Within the group there was a consensus for the approach, which was to use the agile approach. Continuing on this road we wrote the specification on a 1.5 A4 sheet of paper with the design of the game in terms of the end result. This description made it very clear for every group member on what our objective for the development: the developer had to deliver functionality in sprints, with the most value first. The rationale in favour of this approach was to get a better measure of progress, as opposed to a waterfall-like approach, where all requirements are specified up front and handed over to our developer. We thought that a more rigorous process might give us a false promise of progress where a lot of work that should have been addressed earlier becomes apparent near the deadline. Continuously adjusting the process enabled us to find the appropriate amount of formality in controlling the developer. In addition, the continuous adjustments allowed us to find the appropriate degree of formality in communicating the requirements for each sprint that was appropriate for the developer depending on what he understood and the degree of freedom he could handle.

Use of Tools. When managing an outsourced project, we learnt that one should not use tools for the sake of using specific tools. I.e. we set out to use Freedcamp in order to manage our milestones and use it as a communication platform. In hindsight, it was not used a lot, but the project communication did run more smoothly over Skype. We did not have much experience with project management tools and working with Freedcamp was the first for all of us. We mainly used Freedcamp to initialise our project, create a planning and set-up all the necessary to-do's. The programmers were able to look at the entire environment except our to-do's. Both our initial Indian programmers and our Ukrainian developer were added to the project and we tried to use Freedcamp as a communication and planning platform. Both programmers said they had some experience using Freedcamp or similar tools i.e. Basecamp. When looking in hindsight at the project, we can clearly say that the potential of Freedcamp was not completely used. It was mainly used as a repository of sprint specifications and deadlines, which worked quite efficiently.

Real-time collaboration on documents amongst all group members was made possible using Google Drive. Because documents where always 'in sync' at every location, consistency of all requirements was ensured. Skype was used as the main medium of communication with the developers. We used Skype in combination with a small tool called 'Skype Auto Recorder' to record all audio conversations. Not all group members had the time to join the interviews, so using the recordings everybody could listen back to the conversations that had taken place. Quickly, it became clear that it's easier to communicate using Skype's chat function than using audio or video conversations. Both with our Indian as well as our Ukrainian programmer, we experienced low audio quality, and thus decided to communicate using chat. However, we used the video chat option to initialise some sort of bond. MockFlow was used in all sprints to created mock-ups and to illustrate clearly what had to be built. MockFlow is a cloud-based tool to create mock-ups. This tool provides all the necessary elements of a web-page, such as tables and buttons, which can be easily 'dragged and dropped' to a canvas. The free version provided enough functionality to create all the necessary mock-ups.

Project Success. The final product of this outsourcing project was delivered on time and having the functionality specified in the requirements. We qualitatively analysed the delivered software using the ISO 9126 standard. We were confident that the software functionality is sufficient to play the game. All requirements had been incorporated into the application. The outputs of the program were exactly like the requirements stated and as the project group intended. The application had a pleasant layout and beautiful scaling. This increased the attractiveness for users.

For the interactions between the front-end and the server JavaScript was used. The code is based on JQuery, a popular JQuery Library. The specific code written by our programmer had been assessed using JSHint, a community-driven tool to detect errors and potential problems in JavaScript. The results of the JSHint test are shown below (see Fig. 6).

Metric	Value
Lines of code	447
Number of functions	14
Highest number of arguments for a function	2
Median number of arguments for a function	0
Highest number of statements within one function	6
Median number of statements within one function	1
Highest cyclomatic complexity	3
Median of cyclomatic complexity	1
Number of warnings	52 (due to mixing up spaces and tabs)

Fig. 6. Code quality assessment using JSHint (Team3)

Using PHPLoc we were able to analyse the size and structure of the project. Overall the code quality can be classified as good. Since all code was put in classes (except for the code in the index.php, the file to launch the system) we can conclude that the developer adhered well to our requirement of using object-oriented design. Also the average cyclomatic complexity, which is 3.28, shows that most methods have low complexity.

Team 4.

Choosing a Platform and Contractor. For this project we used freelancer.com to find a suitable programmer and Freedcamp to keep track of the progress made. Freelancer is a platform for both developers and project owners. The project owners are able to upload their project, which is then readable for developers. The project owners find and invite developers to look at their project. The developers are able to search for projects and place a bid. Freelancer requires you to upload a project description, the required skills of the programmer and a project name. Additional details can be added using a pdf file. We asked for bids for the entire project regardless of the hours needed. This created a situation where it was the responsibility of the developer to create the software within budget. Additional details were added containing all our requirements.

We were quite satisfied overall with Freelancer, as it got us in touch with the programmer. Despite the relatively short time we needed to find an adequate programmer, there were some major issues to the platform. First of all, upon placing a project we received about eight proposals within the first three minutes. These were all automatically generated and at seemingly random rates. Despite their randomness, they all had 5/5 ratings and admiring comments. However, fake users placed all these comments and ratings. This was tiresome and annoying to say the least, but on a more serious note, it also made us doubt the great ratings that real users had. As a final remark, it should be noted that there are some really strange features in the site. It does not allow you to edit the project name, but you can edit the project description etc. You cannot edit the final deadline, even when you've already handed the project over to a programmer and you have mutually agreed that it should be postponed. This can only be done with the help of the support team. Upon placing a fake project to test the website, it turned out that 'deleting' the project would result in a 5$ fee, while 'closing' the project was free. The only difference between these solutions was the fact that 'deleting' would also make it undiscoverable through the Google search engine.

Contracting. Freelancer takes up part of the money because we had to pay project fees (10 %).

As our project cost was 660$, our programmer only got 600$ of the total amount. Also.

Freelancer kept a percentage (2 %) of the amount of money transferred per transaction. This is something we found out once we paid the first milestone. In the future, to choose a platform we probably would check this first and compare the different platforms on their revenue model.

Specifying Requirements. We made functional, non-functional and technical requirements that were supported by mock-ups, UCD and UML. Not only did we get insights into the difficulty of managing the process of working together at a large distance, but we also got to experience first hand what it took to write real requirements. After an iteration, we gave our programmer feedback as soon as possible. He was very open towards receiving feedback. As soon as he received the feedback the programmer started revising the game. When he was ready he let us know straight away so we could test what he had developed. In the beginning we had some trouble communicating between our group and the programmer, this was mainly due to the fact that we work in a different time zone and the effects of working in a flexible and agile way, as this required more input from our side than we had anticipated. After switching to Skype on mobile, this problem was quickly resolved.

Development Process. Even though many papers would suggest on to go with a fully agile approach for such a small project, we decided we would not. One of the big benefits from agile is the reduced time in documenting, which means increased time for programming. In this project, we were not allowed to contribute to the code, which meant that we had 5 people out of 6 in the team that had plenty of time to do proper documentation. At the start, this meant a very extensive requirements document, which you would not normally find in agile development. The document included information about the entire application, thus it seemed as if we were following the waterfall model, despite of the fact that we had told our developer that we would test what he had made every week and we might change some of our requirements; a rather agile approach. We found multiple benefits of starting the project in a waterfall manner. The first is that it was clear to the programmer what the idea behind each milestone was and where the development was heading. The second advantage was that the programmer was able to agree on a price for the entire program. We have seen in the other student groups that the programmer wanted more money after each sprint, after discovering that the group became dependent of him. They worked it out, but we think the agile sprints did put them in a difficult position. The third advantage was that our programmer worked at unusual hours (weekends and night), which caused problems for providing synchronous feedback. By giving him the entire project in one document, he was able to work at some other part of the software when he was stuck. The next morning we were able to help him, which ensured minimal delay. The last advantage was that our large team of 5 had discussed the entire project before it would start. This minimized discussion and design problems later on and ensured the programmer was able to talk to all of us and get the same answers.

A disadvantage for the waterfall start was that the project needed two weeks before it could be posted on freelancer. We also needed to make sure the developer did not get obsessed with the document too much, and that it was merely a guideline for the situation when he got stuck and when we were not available. We made sure that his progress was visible to us, and that miscommunication or faults in the documents were found quickly. To conclude, we started in a waterfall manner to get clarity on what needed to be done and how much it would cost and from that point onwards we worked agile, to ensure the project was heading in the right direction. We are very glad we did so, because there were many miscommunications at the start and further on in the project and we discovered that the initial document was neither complete nor completely right on some aspects.

Use of Tools. Freedcamp was used to keep track of our progress. Freedcamp can help one monitor one's entire project with posts, to-do's, milestones and files. For informal and fast feedback Skype is the ideal tool, especially in an international working environment. Next to synchronous calls and chat, the chat function also provides asynchronous communication. By using Skype on mobile, both the developer and our team were available every day of the week. We created a new Skype account so that we would not be involved privately. Google Drive was mainly used for file sharing within our team, but some documents (e.g. bug lists and data file examples) were shared with the developer as well. Initially, we started using Freedcamp for setting milestones and to-do lists. However, later on we shifted to Skype and Google Drive for the communication and file sharing. Although we see the positive aspects of Freedcamp when working in larger projects, in our case we basically worked with the one external programmer. Within our group we communicated through WhatsApp and one of us communicated the outcome of our discussion to the programmer.

The group members, who were available when the code was delivered, performed the testing. One of us logged in as the administrator and invited the rest to play the game. While playing the game we mentioned it to each other on Skype if we encountered bugs. We all were listing our bugs in a Google spreadsheet that was shared with the programmer later on.

Project Success. The software worked well for its purpose, but could be designed and documented better. For checking syntax we have use the tool PHP Code Sniffer. This command line tool analyses all PHP, CSS and JS files and checks the syntax. Most errors and warnings are because of syntactical errors, nothing too serious. Furthermore the level of documentation is not very high which might cause some trouble in the future when maintaining or extending this tool.

4 Conclusions and Discussion

While E-markets for global sourcing of projects have been established for over a decade and more recently the market has consolidated by mergers of several leading players, the phenomena is still under researched and its potential to source small and medium sized software development projects is being debated. Extant research has mainly focused on macro characteristics of E-markets. Detailed insight into successful practices for developing software using E-markets is lagging. In this research we aim to provide more insights into the potential of E-markets to successfully source software development projects. Through a field experiment in which four teams develop a software product using identical initial requirements (budget and time schedule, but are otherwise free to select an E-market, tooling, development process and vendor), we aim to gain more insight in factors that determine success and the role of the E-market to facilitate this success.

The research model developed by [9] provided a useful framework for our analysis. Overall, our findings suggest that E-markets can be used to successfully develop a small sized software system in quite spectacular timeframe and very limited budget. Three out of four teams managed to have a working product according to the initial requirements specifications. This result exceeded our expectations and contradicts the critical observations and opinions in several blogs and news articles.

From the detailed experience reports and log files of the four teams several lessons can be drawn that can aid buyers on E-markets to achieve success:

Choosing a Platform and Contractor: All platforms selected by the teams (Odesk, Guru and Freelancer) did fulfil their key features. Still, a key issue is that too many bids are either not legitimate or are automatically generated. Customer ratings seem to be no guarantee. This in-turn makes the selection process cumbersome and time intensive. Additional research on the developers and careful selection are a key requirement for success.

Contracting: The E-markets all offer useful services through predefined payment conditions, where escrow services and guaranteed payments security is provided to both the vendor and customer. However, sometimes hidden fees are applied and also some vendors perceive the fees for each transaction as too high.

Specifying Requirements: Even more than in traditional outsourcing, it appears that having very clear initial requirements and continuous communication about the requirements is key. Cloud based document sharing tools and mock-up tools to visualize screenshot designs are a useful addition. Conceptual models such as UML based activity and class diagrams also proved useful. Non-functional requirements such as performance and code quality are key as well, while these can be specified at a later stage, they have to be clearly communicated and reviewed similar to the review of the functional requirements.

Development Process: A full waterfall nor an agile process did function well for sourcing a small project through an E-market. Carefully mixing the strengths of both approaches worked best, using the waterfall style milestones, precise requirements and contract specifications, combined with more agile rapid adaptation and frequent communications between 'product owners' and developers.

Use of Tools: E-markets have come a long way in supporting more than just vendor search, contracting and payment, and also offer communication tools and project management support. Still, several other tools for communications, requirements specification, modelling and code quality tests proved their value in reaching success, as shown in previous research [10]. Cloud based tools to draw mock-ups greatly enhanced communications. Use of video demo movies communicated intermediate results between the remote and on-site teams. Skype was frequently used especially its basic chat functionality. Remote access to the development server requires high mutual trust, but allowed for frequent testing and feedback. Finally, the use of code quality assessment tools enabled the customer team to assess code quality objectively.

Project Success: It is important to adopt a balanced view towards product success. Usage of standards, like ISO 9126, facilitates a balanced view on product success, including non-functional product quality. Even assessment of comments in the source code should not be forgotten. When the customer has a tight budget and schedule, the focus can be easily drawn to deliver functionality in time. By dividing roles at the customer site to assess different quality aspects, the variety of quality attributes can be guarded.

Gary Swart, CEO of Odesk writes on the company's website communication on the Odesk-Elance vision of the new combined company: "Just as Amazon reinvented retail, and Apple iTunes transformed the music industry, together oDesk and Elance will revolutionize the way we work. This merger will create unprecedented freedom for people to find job opportunities regardless of their location, and will allow businesses of all sizes to more easily access the best available talent".

Indeed our research confirms that it is feasible to run small sized systems development projects successfully through E-markets like Odesk-Elance. However, several factors need to be taken into account and a skilled customer team with competences including vendor selection, software contracting, software requirements specification, development methods, cross cultural and virtual communications, use of various cloud based tools, frequent functional and non functional testing are necessary. Other factors such as socio-technical coordination [11, 12], also have to be taken into account for large projects with multiple vendors or a vendor company employing multiple developers. By no means is the use of these E-markets a guarantor for success. Future research could focus on collecting more experiences and developing guidelines for enhancing the chances of success for small and also larger development projects.

Acknowledgements. The authors would like to express their gratitude to the dedicated work of the software vendors quoted in this paper and the skilled efforts and detailed high quality reporting by the four project teams of class 2014 of the Global Project Management Course at the University of Twente.

References

1. Sakthivel, S.: Managing risk in offshore systems development. Commun. ACM **50**, 69–75 (2007)
2. Persson, J.S., Mathiassen, L., Boeg, J., Madsen, T.S., Steinson, F.: Managing risks in distributed software projects: an integrative framework. IEEE Trans. Eng. Manag. **56**, 508–532 (2009)
3. Amrit, C.: Coordination in software development: the problem of task allocation. In: ACM SIGSOFT Software Engineering Notes, pp. 1–7. ACM, (Year)
4. http://techcrunch.com/2014/11/25/elance-odesk-30m-benchmark/
5. Anders, G.: Why Freelancers Are Dismayed At Elance's Merger With oDes (2013)
6. Assemi, B., Schlagwein, D.: Profile information and business outcomes of providers in electronic service marketplaces: an empirical investigation. In: Proceedings of the 23rd Australasian Conference on Information Systems 2012, ACIS 2012, pp. 1–10. ACIS (Year)
7. Aguinis, H., Lawal, S.O.: eLancing: a review and research agenda for bridging the science–practice gap. Hum. Resour. Manag. Rev. **23**, 6–17 (2013)
8. Gefen, D., Carmel, E.: Is the world really flat? a look at offshoring at an online programming marketplace. MIS Q. **32**, 367–384 (2008)
9. Walter, A.: Success Factors in Leveraging Freelance Marketplaces in Software Development Projects. University of Ottawa (2013)
10. Katsma, C., Amrit, C., van Hillegersberg, J., Sikkel, K.: Can agile software tools bring the benefits of a task board to globally distributed teams? In: Oshri, I., Kotlarsky, J., Willcocks, L.P. (eds.) Global Sourcing 2013. LNBIP, vol. 163, pp. 163–179. Springer, Heidelberg (2013)
11. Amrit, C., van Hillegersberg, J.: Coordination implications of software coupling in open source projects. In: Ågerfalk, P., Boldyreff, C., González-Barahona, J.M., Madey, G.R., Noll, J. (eds.) OSS 2010. IFIP AICT, vol. 319, pp. 314–321. Springer, Heidelberg (2010)
12. Amrit, C., van Hillegersberg, J., Kumar, K.: Identifying coordination problems in software development: finding mismatches between software and project team structures. arXiv preprint. (2012) arXiv:1201.4142