Journal of Information Technology Education: Discussion Cases

11/28/ 2013

MANISH AGRAWAL, GRANDON GILL

MOBILE APPLICATION DEVELOPMENT STRATEGY AT VOLOGY1

When our customers come back, it is for price and speed of delivery

John Dionne, Director of software development at Vology, one of the fastest-growing companies in the Tampa Bay area, had been thinking hard the last few days about different options for developing mobile applications. While he was proud of his existing application portfolio and the way his team supported the company's business, it seemed that all everybody – customers, suppliers, even internal employees – wanted from him was mobile apps. And this had been driving most of his thinking and research the last few weeks.

The sudden emergence of a persistent demand for mobile applications from his base of generally satisfied users had initially taken John by surprise. He knew he was doing a good job. His company's business was growing much faster than the rest of the economy, and his team was supporting this growth with minimal application downtime. However, after researching the landscape, he could sympathize with his users. Seemingly overnight, many factors had converged to render obsolete many of his assumptions about technology. For one, general purpose smart-phones were now more functional than many special-purpose hardware devices. As a result, for the first time that he had experienced in his life, users carried more sophisticated personal devices than the company-issued devices. Naturally, users found it very convenient to do as much work as possible using one device – their own personal device. Having understood this dynamic, he realized that the demand for mobile applications was no passing fad, and it was in his interests to respond to the demand and "get with the times."

But, since the domain was so new, there had not yet emerged in the industry any consensus on "best practices" for developing mobile applications. He could wait until some best practices and standards had emerged. But by then, at least some competitors would have developed something and gotten an edge over Vology. Or, he could take the risk of trying to figure out things on his own, making mistakes along the way, and paying for them from his own limited budget. Didn't someone say, "*experience keeps a dear school, but fools will learn from no other*"?

Based on his discussions with industry professionals, he seemed to have three major options: (1) use HTML/ JQuery to create web applications; (2) use platform-specific developer tools to create platform-specific applications; (3) use RAD tools such as Sencha and RhoMobile to create platform-neutral web-based applications.

[1] Copyright © 2013, *Informing Science Institute*. This case was prepared for the purpose of class discussion, and not to illustrate the effective or ineffective handling of an administrative situation. Names and some information have been disguised. Permission is granted to copy and distribute this case for non-commercial purposes, in both printed and electronic formats.

Vology

Vology was founded in 2000 when its founder, Barry Shevlin, sensed opportunity in reselling barely used computing and networking equipment owned by failed companies in the aftermath of the dot-com boom. In 2002, the company hit the $1 million revenue mark. This number had grown to almost $100 million in 2012, an annual growth rate of over 55%, sustained for over a decade. Even more impressive was the fact that this growth had been accomplished in an environment where the US and global economies were generally weak, including deep recessions in many of Vology's largest markets in the US and Europe during 2008-2012. Leveraging technology to streamline business processes had been central to the company's ability to manage this rapid growth during its first decade in business.

Vology characterized its offering as a Hybrid SuperVAR. VARs are value-added resellers, who add value to existing products through features or services. The product, integrated with the added features and services, is sold to end-users. VARs are common in industries where substantial local expertise and effort is needed to identify the right components to create a complete solution. For example, when companies create a computer network, they need to buy and install switches and routers, cables, and other hardware. To identify the right components, they need to anticipate current and future needs and may have to integrate products from multiple companies to create the solution that best fits the needs of the customer. Value added resellers have the incentives and resources to maintain local expertise to meet user needs. VARs are the standard sales channel in the electronics industry. Equipment manufacturers often find it more profitable to train and support VARs than to maintain their own sales channels. Vology was a VAR or independent reseller for most of the top equipment manufacturers in the industry, including Cisco, Juniper, Brocade, and HP.

While there is no standard definition for a SuperVARs, a company that fits the description has a national or international presence and sizeable revenues to be able to offer a wide range of services to its customers. $100mn in revenues is the typical benchmark for a company to be recognized as a SuperVAR.

Hybrid VARs combine traditional VAR services (selling new equipment along with added services) with certified pre-owned products. A hybrid VAR can help organizations maximize the value of their IT infrastructure budgets by acquiring used products where appropriate along with new equipment. Hybrid VARs such as Vology take away the complexity of dealing with multiple vendors – VARs and used equipment sellers.

Middlemen

Businesses such as VARs are considered middlemen or intermediaries. Intermediaries, such as retailers, purchase goods from suppliers and resell these goods to buyers. Other intermediaries, such as auction houses, help buyers and sellers find each other. Intermediaries generate economic value by identifying reliable suppliers, finding buyers for the products sold by these suppliers, determining prices for buying and selling, providing commercial services such as payments and record keeping, and maintaining inventories of products to provide liquidity or availability of goods and services. Managers at intermediaries have to make decisions on the mix of products they will purchase from suppliers, the types of suppliers they will deal with, and any additional services they will offer to their customers.

Inventory strategy

Most customers value expedited fulfillment of orders. Therefore, at any given time, Vology maintained an inventory of pre-owned equipment with a retail value of almost $50 million. At the time of writing the case, CISCO 3500 switches and CISCO 7000 phones were among the most popular items among customers. Vology went to extra lengths to ensure that these high volume items were always in stock. At any given time, the firm had almost 7,000 items in its inventory, including many products from HP and CISCO which, being popular, were always in stock.

However, since technology equipment becomes obsolete in months, the company is always under pressure to turn around its existing inventory. Superior business processes helped the company manage this large inventory profitably. One measure of the success of the company was that for some years now, it was the second largest seller of pre-owned Cisco equipment in the world.

Inventory handling

Handling the inventory involved essentially three activities– intake, put-away, and pick-up.

Intake processes incoming items. Every day, Vology receives trucks full of items it has purchased from other vendors. These goods come from various sources such as companies upgrading or restructuring their IT infrastructure, overstocks, and even companies going out of business. To facilitate processing, a bin is created to gather all the items related to one order. The purchase order is affixed to the bin and the hand-held device is used to quickly obtain the part and serial numbers of the different items. The bar-code scanning features of the hand-held device saved the effort of manually entering item information. A label identifying the part is then printed from the hand-held device and affixed to the part for subsequent identification. When all the items from an order are collected, the intake process completes and the bin is moved to the testing area where each part is tested before the bin is moved to the next activity of the inventory handling process – put away. Exhibit 1 shows some activities in the process. Exhibit 2 shows the testing area at Vology.

As the name suggests, put away stores items in inventory after they have been rigorously tested. If other items with the same part number exist in inventory, then the item is stored along with the other items of the same type. If not, a new location is created for the item in the warehouse. While creating the new location, the warehouse staff has to optimize space utilization and order processing. The more popular items have to be kept in a location where they are easier to drop off and pick up. The less popular items can be stored in relatively inaccessible areas. As each item is put away, workers use their hand-held devices to record the exact locations of each item. Exhibit 3 shows a section of the inventory in the warehouse.

Pick-up is the activity used to complete a sales order. When an order comes in, the inventory management system lists out all the items in the order and their location in the warehouse. A warehouse worker then gathers all the items from the warehouse and prepares them for shipment.

Industry

From Vology's perspective, the industry was composed of about 300 firms nationwide. These firms sold used, surplus, and pre-owned equipment, a lot of which was in fact unused and unopened. Firms within the industry were ranked on the basis of sales, breadth of product lines offered, transactions volume, industry segments served (networking, storage, computing etc), and sales reach (local/ national/ international).

All firms in the industry compete to be the single source for all their customer's equipment needs. Repeat business in the industry is typically very high. Vology's operating principle for satisfying customers is, "*when our customers come back, it is for price and speed.*" Vology's estimate was that about 60% of its revenues came from pre-owned equipment and about 40% from new equipment, sourced directly from an authorized distributor such as Tech Data or Ingram. Networking equipment (switches, routers, phones) is the most popular category in the industry.

Pre-owned gear is typically sourced from firms going out of business. Also, surprisingly, a lot of equipment is sourced from competitors. There is a high level of co-operation among the serious players in the industry, in part because every player has a strong incentive to move their inventory as rapidly as possible and to hope for co-operation in return to meet their own customer orders.

UNEDA

The players in the industry have an interest in maintaining the viability of the pre-owned gear market, particularly in ensuring that the equipment they sell is of the highest quality. Without that assurance, most customers would not even consider the channel to meet their requirements. Accordingly, the industry has created the United Network Equipment Dealer Association (UNEDA), a worldwide alliance of more than 300 of the leading marketers of pre-owned networking equipment. While the organization was formally established in 2006, many companies in the industry had already been partnering with each other and working towards maintaining equipment quality. UNEDA members agree to uphold strict guidelines for ethical business practices and to ensure that buying pre-owned networking equipment remains a cost-effective and viable alternative to purchasing new networking equipment from manufacturers or their authorized distributors. To accomplish this, UNEDA's coalition eradicates the presence of counterfeit, fraudulent and stolen gear in the secondary market.

The Order-Handling System

Not one to shirk a challenge, and being part of an entrepreneurial, high-growth company with great top management support for sensible risks, John was already beginning to sense the excitement awaiting him if he chose to learn his way through the landscape of mobile application development. What tools to use, what languages to use, what training to provide, what form factors to support. He was convinced it was not foolish of him in this case to figure all these things out from personal experience.

After considering many alternatives, John had decided that his first mobile application development effort would relate to the company’s order-handling system. This application was one of the most critical to the company’s business. He figured that introducing a mobile interface to this application would get him very high return on investment as well as top management support. This support would be crucial in the early stages to deal with unanticipated issues.

The order-handling system was Vology’s primary enterprise IT system. John planned to introduce mobile interfaces to selected parts of this system. Exhibit 4 provides an overview of Vology’s order handling system. The system had three main components – a customer-facing e-commerce system, a back-end system performing the core enterprise operations, and an accounting system.

The e-commerce system

Vology had just deployed its brand-new e-commerce system to replace its home grown NOP CRM system. The new system was a customized implementation of the open source Broadleaf e-commerce system, which replaced an older ASP-based e-commerce system. Traditionally, Vology’s sales were human-intensive. Customers called the central sales number, where a member of the sales staff went through the customer’s needs, identified the right product, and promptly sent a quote. Specifically, Vology currently had no web-based ordering system. This is in-part because its customers did not typically pay by credit card. Payments were generally handled by the purchasing departments of companies who paid by check.

However, since salesperson time is very expensive, Vology could not really afford for sales people to handle calls for low-priced parts such as cables. Vology wanted a system whereby customers for these simple products could self-help and place orders directly. The company had given considerable thought to the impact of this change on the commissions earned by its salespeople. Vology did not want to adversely impact its salespeople, since they brought in the bulk of the company's revenues and developed new client relationships. However, the leadership at the company was convinced that the new system would actually help salespeople by reducing the time they spent dealing with low revenue calls. In addition, any e-commerce system at the company would have order thresholds. Orders totaling more than the threshold would be directed to a live salesperson for configuration, validation, etc. This, John thought, would ensure that the company sales force was not hurt by the web ordering system.

When Vology had decided to replace its e-commerce system, it engaged a consulting company, Credera. These consultants examined Vology's current system and assessed where the company wanted to be in the next 5 to 10 years. Vology's vision was to be above a generic cable seller, but it also did not immediately anticipate being a serious competitor to a major VAR such as CDW. Based on this vision of the company, the consultants identified the main requirements for Vology's e-commerce system as (1) search engine optimization (SEO), (2) improved user interaction, and (3) lead generation.

With the assistance of the consultants, Vology's software team evaluated e-commerce solutions from Oracle and Microsoft as well as Broadleaf . Although John had limited familiarity with Broadleaf, the solution came highly recommended from the consultants at Credera. Some of the features that worked in Broadleaf's favor included Broadleaf's open-source heritage, and its use of a standard Java web framework - Spring MVC.

Broadleaf e-commerce

As an e-commerce platform, Broadleaf included all the standard features such as a product catalog, the capability to run promotions, and displaying targeted advertisements based on the criteria satisfied by the customer. The Core Broadleaf system is built around three entities that are critical to any e-commerce system – product, SKU, and order (cart).

Customer interactions in the Broadleaf e-commerce system are handled through workflows and activities. Workflows respond to common end-user needs, for example adding items to carts, pricing orders, and checking out orders. Workflows are built out of activities. Activities in Broadleaf are units of action and error handling. To help customers find the products they need, Broadleaf has a powerful search and browse capability. Broadleaf even allows companies to implement dynamic pricing, which is the practice of specifying prices based on the properties of each individual request instead of simply providing a uniform pricing for a given product to everybody. While this practice is controversial, location-based pricing can be implemented using the Broadleaf e-commerce system. More details about Broadleaf (including the source code to the entire system for those interested) is available at the product's website[2].

Broadleaf's initial database configuration uses the Hypersonic (HSQL) database. Exhibit 5 provides an overview of the Hypersonic database. However, Broadleaf can also be configured to use any of the popular database servers including Microsoft SQL Server, Oracle DB and the two popular open source databases – MySQL and Postgres. The initial application server used by Broadleaf is JeTTY. However keeping with the promise of Java's portability, Broadleaf is application server agnostic and can run on any Java application server, including Tomcat. Broadleaf also helped firms maintain PCI compliance in case they chose to store sensitive customer financial information.

As seen in Exhibit 4, Vology's vision is for the eCommerce system to be the common order-taking system for retail customers, with potentially adding functionality at a later phase to allow sales people to cre-

[2] http://www.broadleafcommerce.org/

ate orders for customers using a similar interface. The e-commerce platform will also have business-friendly features such as integrated chat and technical support, as well as other features for a customer to interact with their account manager or required Vology employee.

The back-end system

The eCommerce system interfaces with Vology's home-grown back-end system. This system is responsible for all operations at the company including CRM, Accounting Integration, and inventory management. Vology's appellation for this system was SMART. SMART also continues to be the system of record, maintaining information regarding commissions, sales reporting, customer accounts, and customer leads.

Development of SMART had begun in 2004 with .NET version 1.1, but over the years, the application had leveraged advances in the .NET 4.0 platform and currently used the 4.0 version of .NET.

Details of all business transactions, such as orders and receipts, were stored in SMART. The software in the handheld devices was the inventory management subset of the SMART software, which the IT team called SMART inventory.

The accounting system

Vology's accounting system was built on Microsoft's Great Plains accounting software. One of the functionalities provided by SMART was to provide an interface between the CRM system and the accounting system.

Users

John could see two distinct categories of users of the system. Within the firm were the warehouse staff who could use the application to manage inventory. External to the firm were customers who may like to monitor the progress of their orders, make quick enquiries about order status, call their account executive etc.

The Mobile Application Environment

Vology was interested in creating mobile interfaces for two applications in its order-handling system – the eCommerce system and the SMART system. As may be inferred from Exhibit 1, Vology currently had no mobile capabilities for its recently released eCommerce site.

The existing mobile functionality at the company was limited to proprietary hand-held devices that allowed its warehouse members to perform standard warehousing functions based on reading bar-codes. Over the years, these devices had improved. The first lot he purchased in 2005 did not have non-volatile storage. If the batteries ever ran out, so did the application. When that happened, it took about 15 minutes to reload the inventory management application to the device. The new lot he purchased in 2010 had Compact-Flash (CF) storage, so it was not necessary to reload the application after every recharge. The UI of the hand-held devices is shown in Exhibit 6.

The inventory management functions performed using these devices included receiving packages, shipping packages, printing labels for packages to be shipped, and recording transfers of inventory from one location to another – receiving dock to testing area, testing area to warehouse, warehouse to shipping, etc. Thus, the device could only be used for warehousing functions. Accounting, CRM, and other functions could not be completed. This was becoming a serious limitation. Besides, the Windows CE operating system used in the devices was becoming difficult to maintain since Microsoft had moved on to Windows 8 as its primary mobile platform. John was concerned about the implications of this on support for Windows CE from Microsoft, though production support was listed until the year 2022. In general, IT managers are concerned that if the software is no longer supported, newly discovered vulnerabilities would not be patched and the software could raise information security concerns. However, that was not a concern for John. The computer network inside the warehouse was disconnected from the rest of the Internet, so attackers had no path to target these devices. His bigger concern was that he did not want his developers to invest their time developing expertise in a skill set that was destined to be obsolete in the near future. Also, the operating system was becoming primitive by any standards. For example, the only sound the operating system supported was SYSTEM.BEEP.

John also thought it was now possible for his warehouse employees to exponentially improve productivity by adopting a bring your own device (BYOD) philosophy. This could potentially save his division some money since each hand-held scanner unit cost approx. $1,200.

With this background, John wanted to create interfaces to SMART and Broadleaf from the common smart phone platforms – iOS, Android, and, increasingly, Windows 8. Retail customers would use the application to access their account and order items. Sales representatives would use the application to access customer accounts, configure products and solutions, and place orders.

Development models

It did not hurt that there were many classes of users of the order handling system – suppliers, employees, and retail customers. He could start with the user group with the simplest expectations, refine the application to meet their needs, and apply that experience to develop the application further to serve more discriminating users. He could even plan to eventually mix and match applications types to serve different groups of customers, for example, one application for warehouse workers and another application for end users. John's eventual decision would have many elements and would be influenced by a number of factors.

Platform-specific applications

First, there were the competing technology platforms. Apple's iPhone and iOS was obviously dominant. But Android was the most popular platform. And Microsoft had just released its Windows 8 platform to rave reviews. All these platforms supported fancy features such as cameras, accelerometers, gps, etc. The problem was that each of these platforms offered their own developer tools and deployment mechanisms. While there were some commonalities in development languages, there was also a distinct learning curve with each technology.

Even within these platforms, there were some interesting distinctions. While Android was inexpensive, dominant, and even open source, it changed too rapidly for his comfort. New versions were being released within months of the each other. By comparison, Apple's iOS was released at more industry-friendly cycles. As a result, Apple hardware was becoming a commodity. He could get version 2 of the iPAD devices when version 4 had come out in the market. His developers would much prefer developing for the iOS for its relatively predictable evolution, but a lot of his users were likely to be carrying Android devices for their affordability.

Web-based mobile applications

John was obviously not the first to wrestle with the expenses associated with developing multiple versions of essentially the same application, one for each platform – iOS, Android, and Windows. Faced with this issue, the industry had come up with a web-based alternative. He had read that it was possible to develop cross-platform applications using some recently introduced technologies such as HTML 5 and JQuery.

These applications would run on browsers, eliminating the need for platform-specific application development and deployment. If he chose this route, he wouldn't need to develop platform-specific expertise within the company. However, this advantage came with its own costs – web applications do not support many of the rich platform-specific features offered by mobile devices. Also, using HTML5 added a dependence on an additional technology component – HTML5. What would he do if the HTML5 standards changed in a way that affected him adversely?

This was not an issue for John in the short term. Anything he created would be better than his current applications. Nevertheless, he was certain that many new and exciting technologies would become available in the near future, though he couldn't predict their implications and relevance for his business. For example, he hadn't yet found any meaningful business use for the music players built into all these modern phones. Should he be overspending now to be better prepared for unknown opportunities that may emerge in the future?

RAD mobile web applications

Then, in the last few weeks, he read an article in an industry magazine about some cross-platform development tools with names such as Sencha Touch, RhoMobile, and Appcelerator. Apparently, these tools could be used to create native applications using web frameworks such as Ruby. What was going on here?

Based on his reading, these frameworks were Javascript libraries to develop user interfaces specifically for web browsers on Mobile devices. Javascript allows users to use the web browser as a miniature operating system to create applications. The development libraries are used to develop web user interfaces that look and feel like native applications on mobile devices. By automating most of the common development tasks, Javascript libraries such as Sencha Touch speed up application development. The generic industry term for such tools is RAD – rapid application development tools. The primary limitations of these libraries are the limitations of browsers as application development environment.

But he had his concerns with these RAD tools. What if the firm supporting the development of the RAD tool disbands? What if there were some severe bugs in the tools; would the firm have the resources to ensure that the bug would be fixed? What if a chosen RAD tool did not support one critical feature essential to his firm?

These were interesting times, and John was really thrilled to be at the center of the amazing IT industry where the role of people like him was critical to exploiting the potential of technology in improving people's lives through increased productivity, and even "fun."

John had been giving considerable thought to whether to build the application using HTML 5/ JQuery or to have platform-specific applications for each operating system. He recalled industry colleagues telling him that 85% of experts prefer platform-specific applications over web-sites Yet, he couldn't make up his mind one way or the other. "*And now, I have to figure out if I should use HTML 5/ JQuery or go platform-specific. Or try one of these cross-platform tools. If I go platform-specific, is there any platform I can leave out? Should I develop for phones or tablets? What screen sizes? What kind of training would my developers need*," he thought.

Personally, John had a strong preference for platform-specific applications. The interfaces would be superior, more features would be supported, and the general user-experience would be very pleasant. With platform-specific applications, he expected to be able to use third-party hardware such as Bluetooth or USB scanners for added functionality.

By comparison, he expected a HTML 5/ JQuery interface to be generic and inelegant. On the other hand, while the future of any specific OS was unpredictable (witness the rapid descent of the Blackberry), HTML 5 and JQuery were likely to be around forever. This implied that using HTML 5 would not only be economical, requiring only one common web-based application to serve all devices, it would also have a longer life. While some technologists are concerned about the viability of free and open-source technologies such as JQuery, John did not share that concern, at least not at the current time.

Cross-platform web application development environments such as Sencha had potential challenges too. Would these platforms support bar-code scanning using the device's built-in camera? From what he could read, at the time of writing this case, this capability was not integrated into the platform. Though some users seemed to claim on message boards that they had succeeded in using various plug-ins to accomplish such scanning, he wasn't willing to trust unproved software to support a core feature at his company. Furthermore, in his career, he had seen various such promising cross-platform development technologies come, get hyped, but fail to survive. What if such a fate eventually befell Sencha and other tools in this family? Would he discard the application his team had so painstakingly developed?

John believed that most developers prefer cross-platform web development using Javascript to save time and, hence, money. However, the order-handling system was a core application of the company, and saving a fraction of overall development effort was less important to John than getting all the required functionality in the application. For example, he could not afford to compromise on flawless functioning of the bar code scanning functions.

John was also uncertain about the complexities associated with getting the platform-specific applications approved by the application stores managed by the respective platform providers. Web applications needed no such approval since they only used the browser, which was pre-installed on all platforms.

From his review of the competitive landscape, he believed that most of his competitors were navigating towards HTML 5/ JQuery based web applications to achieve the goals that John had for his project.

Design considerations

Recent weeks had added an additional twist to the decision. The distinctions between phones and computers were blurring. Screen sizes were available from anywhere between 5" - 10." Aspect ratios (the ratio of screen width to screen height) were also distributed over as wide a range. It was difficult to create a generic layout that would work for this entire range of screen sizes and aspect ratios. What looked pleasant on one screen could look ugly on another. He wondered how this might impact the usability of his applications and his choice of development environment.

Having given this issue some thought, he realized that the proliferation of form factors was creating an additional decision for him – should he favor the smaller form factors of the phones or the larger form factors of the tablets? The tablets were more informative and allowed developers to squeeze more information on one screen. The applications could show images alongside text for quick item identification. However, the phones were more portable. They could be worn on the back of the palms for quick order review. Workers would be able to use both hands to haul merchandise. On the inventory floor, this could be a big deal. At this time John wasn't sure which was the bigger deal – portability or information availability.

Development team

John software development group was composed of two teams. One team was responsible for SMART. This team had two developers and another developer who worked part-time. SMART used the C# programming language, and this team had built considerable expertise in C# and related technologies. The other team was responsible for the eCommerce site. This team had four developers who had earlier been working on Microsoft's .NET technologies. This team had built and supported Vology's ASP-based eCommerce web site. They had also done all the NOP development. When this site was replaced by the Java-based Broadleaf solution, the team had switched to Java. After an initial transition period, this team now had strong Java expertise.

A year ago, his team had abstracted all hard-wired communication between the SMART application and the credit card processing service (Cyber source) and moved them to web services. This project helped the team in two ways: (1) it gave them experience in using web services and the APIs involved in creating and consuming web services; (2) it enabled Vology to replace the old eCommerce system with the Broadleaf platform. John found it particularly satisfying that his team could get the Java-Spring MVC based website to communicate seamlessly with the .NET-based SMART system. The transition is shown in Exhibit 7.

Critical to the successful transition was identifying all instances of tight coupling between SMART and CyberSource and converting them to loose couplings between the systems. Coupling is an important element of software design and at the two ends of the coupling spectrum are "loose coupling" and "tight coupling." Coupling among subsystems reflects the extent to which subsystems depend upon a knowledge of the internal details of other subsystems. Loosely coupled sub-systems depend only upon knowledge of the outputs of other sub-systems. While tightly coupled systems appear easy to develop in the early stages of a project, over time tightly coupled systems become difficult to maintain. This is because changes in the internal workings of one sub-system can require changes in many other sub-systems even if the final result of the sub-system remains the same. Therefore, it is considered essential in commercial software development to design loosely coupled software.

The successful transition of this project to a loosely coupled system gave John further reason to feel confident in the ability of his team to meet the challenge of developing mobile applications using technologies currently unfamiliar to the team.

John had given some thought to hiring a professional services firm to develop the application for them, based on the requirements specified by John and his team. However, Vology had a strong DIY (do-it-yourself) culture. In particular, John greatly appreciated that the CEO, Barry Shevlin, shared his philosophy of trying to keep the firm self-sufficient to the extent possible. In fact, one of Barry's favorite personal quotes was "*show us how to hook the worm, but we will fish on our own*." His site was operational 24 hours a day, but if something broke at 3am, he was more confident of his ability to get his own team to respond to the problem promptly than in his ability to get a consultant to attend to the problem.

All testing was done by this team. Interestingly, while automated testing procedures are all the rage at industry conferences, John found that most testing at his firm was actually human-intensive. In his experience, the Java side of his environment used more automated testing procedures than the .NET side.

Staffing

Adding staff to assist with the mobile application development was on John's wish-list. However, John was aware that no hiring was imminent at his firm. He would have to depend on his existing staff to take on the application development responsibilities, in addition to their current job responsibilities. He did anticipate his staff facing a steep learning curve, but his staff handled the transition from C# to Java in about 1 month. "*The transition to mobile development can't be much greater,*" he thought. After all, Android development is just Java development. At the time of writing the case, one developer was using his spare time to learn Objective C, the language used to develop applications for Apple products such as the iPad and iPhone. If any personnel needs emerged, he anticipated going the contractor route, hiring experts temporarily to fill in immediate needs.

Project estimates

John had drawn up some simple use cases and activity diagrams to estimate the complexity of the project (Exhibit 8 and Exhibit 9). He had also been collecting some information to estimate the traffic volume he could anticipate coming in from the mobile application. This information was useful to know in order to scale the back-end servers that would respond to user queries.

Within the company, of the current 160 employees, about 100 were directly involved in sales and warehousing. He expected all of these 100 employees to be serious users of the system, performing purchasing, sales, and warehousing operations using the application. It was difficult to anticipate the transaction volume from the public website, but he figured it would be comparable to the volume generated by internal users, at least at first.

Current status

John was finding it difficult to choose between his three major options: (1) HTML/ JQuery to create web applications; (2) platform-specific developer tools to create platform-specific applications; (3) RAD tools such as Sencha and RhoMobile to create platform-neutral web-based applications. All had their pros and cons and none emerged the clear winner yet.

John currently anticipated the development to get serious in the second half of the next year. He had budgeted for some Apple Mac computers for his developers to work on Apple software. He was planning to run a couple of pilots to let his developers get their hands dirty, learn the work flow of developing a mobile application, and get valuable user feedback on functionality. With this input, his team would be able to get to the first version of the application in about 3 months, or so he thought.

Future directions

While he was debating his development options, he couldn't help but note that technology development continued apace. Siri and Google glass were on the horizon. He could visualize a future where warehouse workers wearing Google Glass could scan a bar code by just looking at the tag. The user would receive visual and spoken directions to the bin in the warehouse. Apple's competing voice-recognition technology, Siri, held similar promise. Perhaps he might find a way to keep his application future-proof so that these technologies could also be integrated into the application when they were ready for prime-time, perhaps in a couple of years.

Additional Reading

Fowler, M. (2003). *UML distilled* (3rd ed). Addison-Wesley.

Miller, J. (2008, October). Cohesion and coupling. *MSDN magazine*. Retrieved August 24, 2013, from http://msdn.microsoft.com/en-us/magazine/cc947917.aspx

Schneider, G., & Winters, J. P. (2010). *Applying use cases: A practical guide*. Pearson. Retrieved February 5, 2013 from http://www.ibm.com/developerworks/rational/library/content/legacy/parttwo/1000/0670/0670_Schneider_Ch07.pdf

Spulber, D. F. (1996). Market microstructure and intermediation. *Journal of Economic Perspectives, 10*(3), 135-152.

Exhibit 1: Intake Process

Incoming item information being gathered using the hand-held device

Item next to bin in which it will be placed

Exhibit 2: Testing Area at Vology

Exhibit 3: A Section of the Warehouse at Vology

Exhibit 4: Vology's Order-Handling System

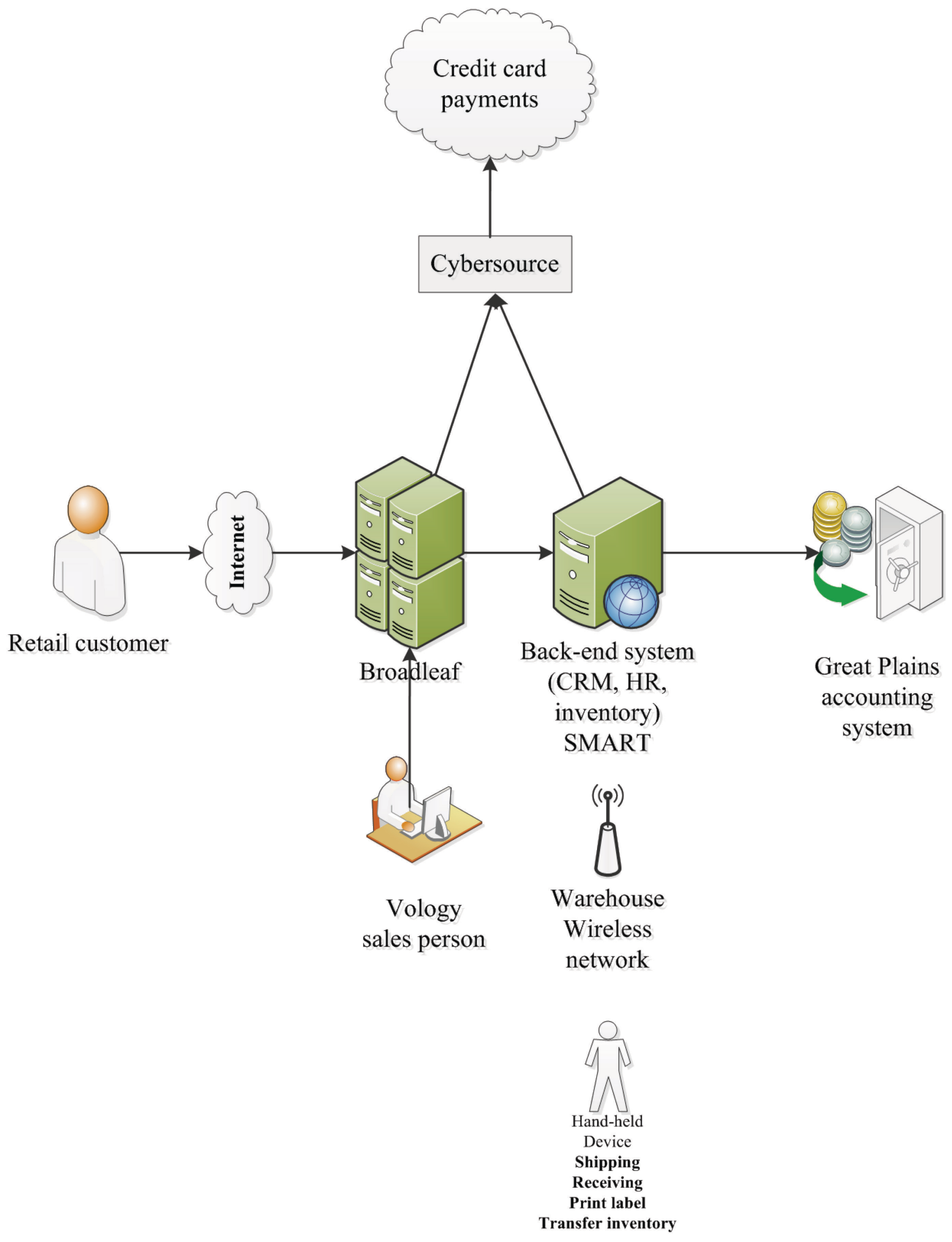


Exhibit 5: The Hypersonic Database

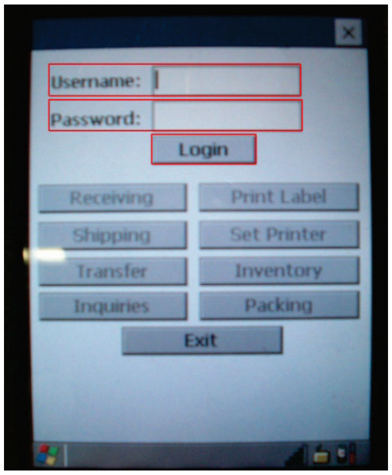
HSQLDB (Hyper Structured Query Language Database) is an open source relational database management system written in Java. According to its website, its benefits are that it is small, fast and multithreaded. It supports most features specified by database standards. For ease of use, HSQLDB has additional features including a simple web server and web-based management tools.

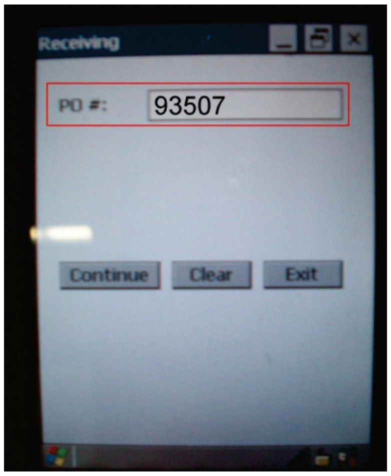
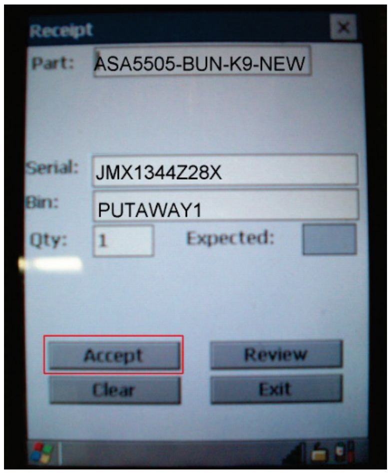
HSQLDB is used to provide storage features in many open source software projects, including OpenOffice.

More information about the Hypersonic database can be found at http://hsqldb.org

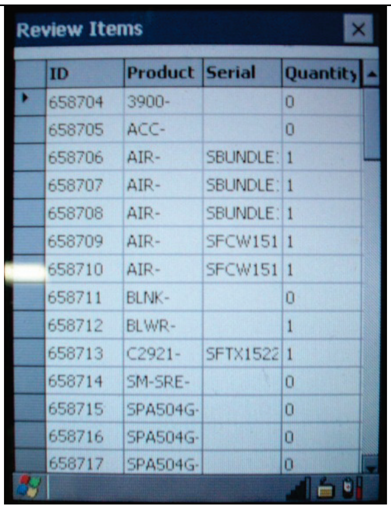
Exhibit 6: UI of Current Hand-Held

Main menu



Intake – Enter PO	**Intake – Item information scanned**
 Receiving PO #: 93507 	 Receipt Part: ASA5505-BUN-K9-NEW Serial: JMX1344Z28X Bin: PUTAWAY1 Qty: 1 Expected:

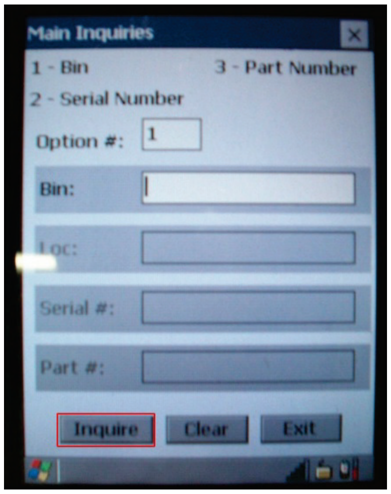
Intake – Order review



Review Items

	ID	Product	Serial	Quantity
▸	658704	3900-		0
	658705	ACC-		0
	658706	AIR-	SBUNDLE:	1
	658707	AIR-	SBUNDLE:	1
	658708	AIR-	SBUNDLE:	1
	658709	AIR-	SFCW151	1
	658710	AIR-	SFCW151	1
	658711	BLNK-		0
	658712	BLWR-		1
	658713	C2921-	SFTX1522	1
	658714	SM-SRE-		0
	658715	SPA504G-		0
	658716	SPA504G-		0
	658717	SPA504G-		0

Put away – Search query

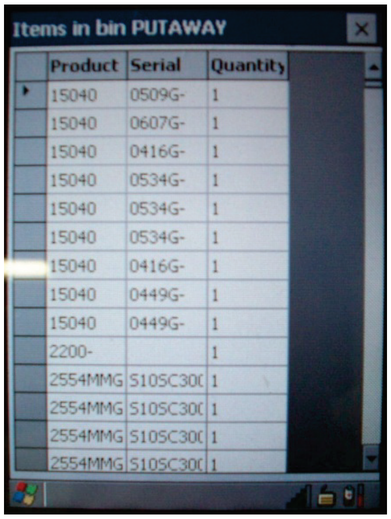


Main Inquiries

1 - Bin 3 - Part Number
2 - Serial Number

Option #:	1
Bin:	
Loc:	
Serial #:	
Part #:	

Put away – Search results



Items in bin PUTAWAY

	Product	Serial	Quantity
▸	15040	0509G-	1
	15040	0607G-	1
	15040	0416G-	1
	15040	0534G-	1
	15040	0534G-	1
	15040	0534G-	1
	15040	0416G-	1
	15040	0449G-	1
	15040	0449G-	1
	2200-		1
	2554MMG	S10SC30(	1
	2554MMG	S10SC30(	1
	2554MMG	S10SC30(	1
	2554MMG	S10SC30(	1

Exhibit 7: Vology's Web Services Transition

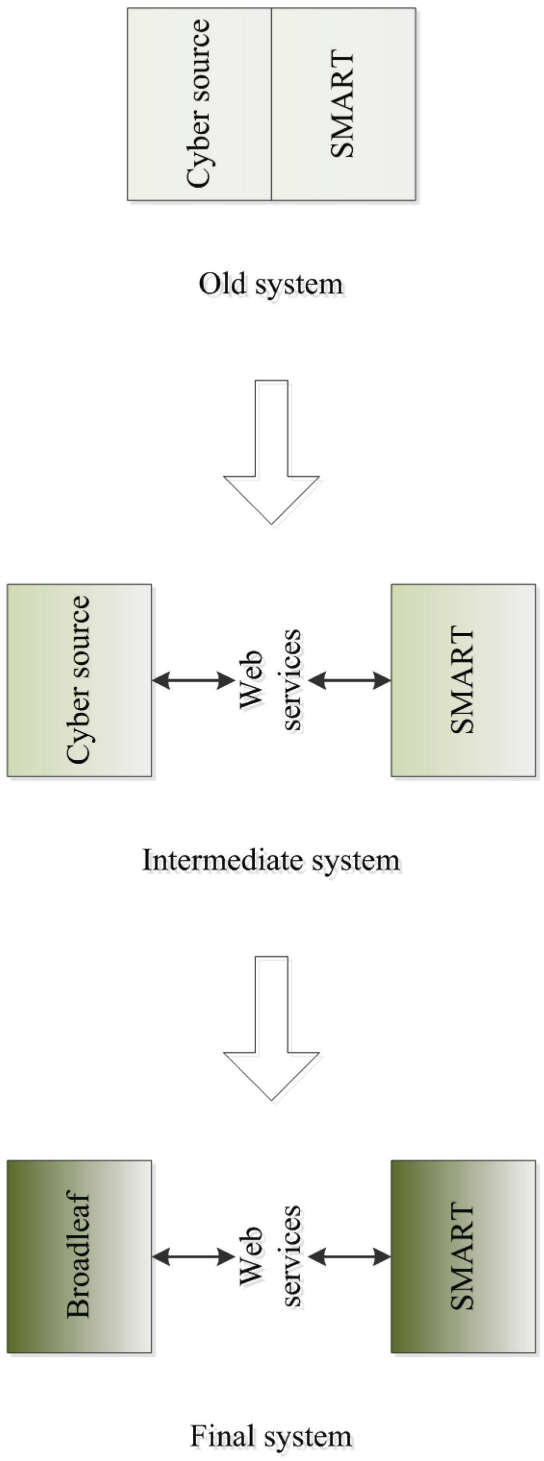


Exhibit 8: Use Case Diagram for Order Processing

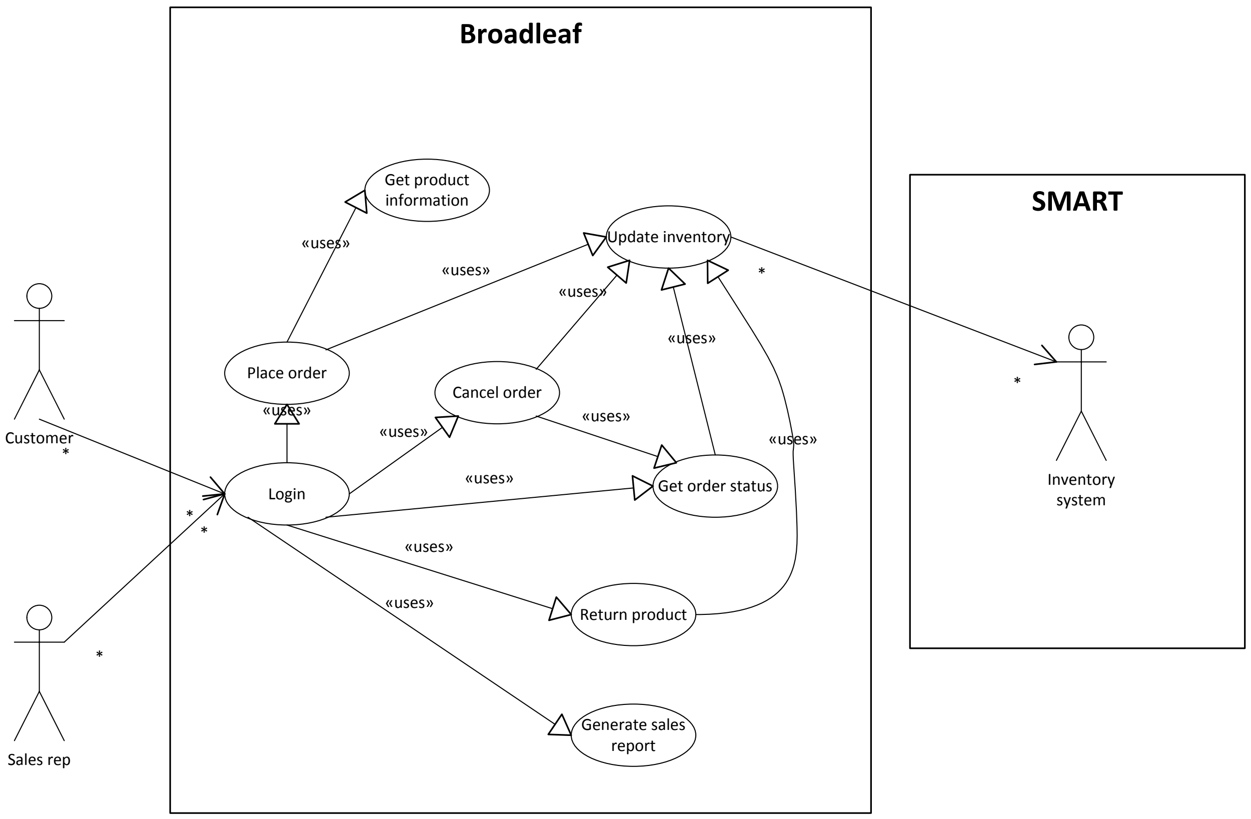
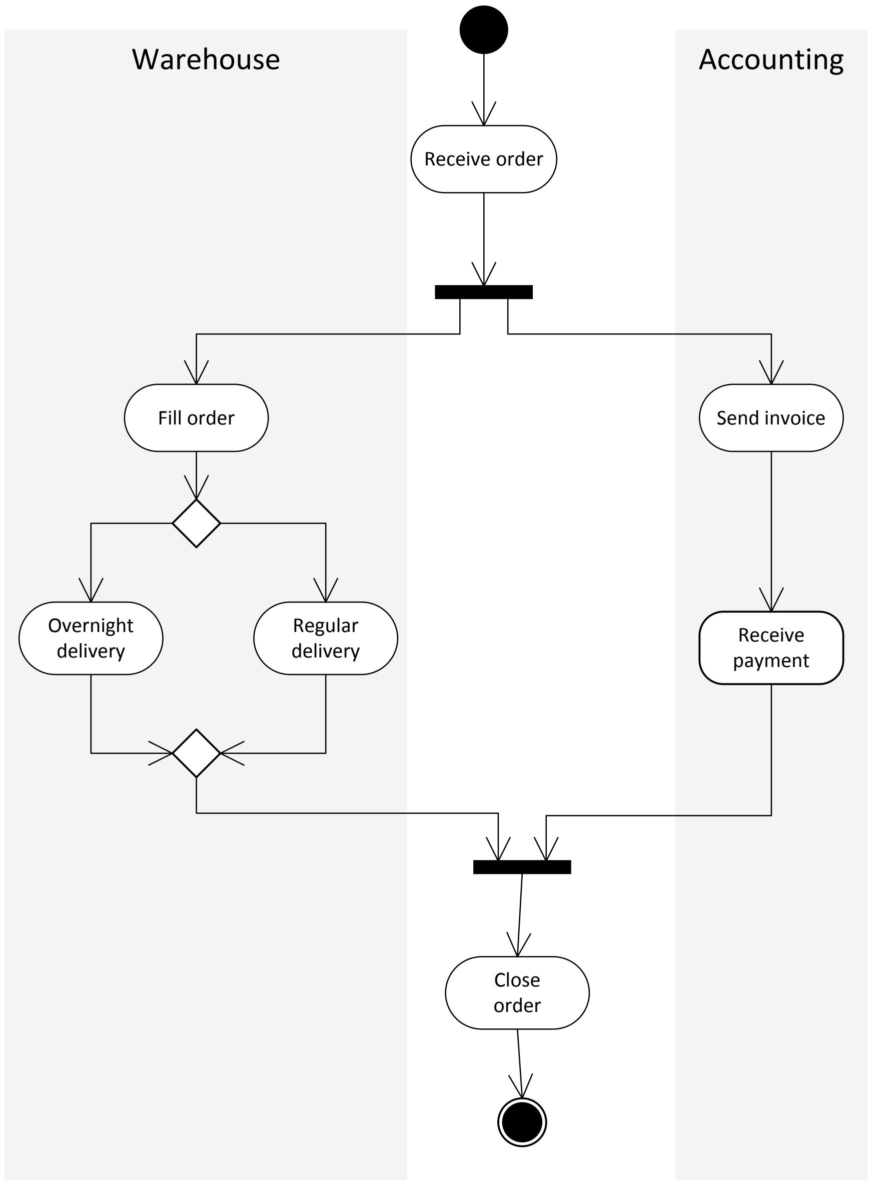


Exhibit 9: Activity Diagram for Receiving an Order


Warehouse
Accounting
Receive order
Fill order
Send invoice
Overnight delivery
Regular delivery
Receive payment
Close order