Richard Ivey School of Business
The University of Western Ontario

IVEY

W11744

PEARSON'S SUCCESSMAKER: PUTTING THE CUSTOMER FIRST IN TRANSFORMING PRODUCT DEVELOPMENT PROCESSES

T. S. Raghu and Collin Sellman wrote this case solely to provide material for class discussion. The authors do not intend to illustrate either effective or ineffective handling of a managerial situation. The authors may have disguised certain names and other identifying information to protect confidentiality.

"Complacency Kills" was circled and underlined in Greg Adams-Woodford's notes. It was the image of those words that had stuck with him over the weekend. Now it was Monday and, as the vice-president of product management, it was time to determine the product development road map for the next five years. It was hard for Adams-Woodford to believe it had only been a year since he first joined Pearson. He was proud of the positive impact his work had on improving the market position of the newest version of the SuccessMaker product in such a short span of time.

When Adams-Woodford first joined Pearson, the entire digital learning division was in the midst of the first major upgrade of its SuccessMaker product. SuccessMaker was Pearson's premier software product for helping elementary and middle school students to improve their math and reading skills, so this upgrade was a strategically important project for the firm. The upgrade involved significant changes to the foundational technology of the product — from an older client-server model to a new Internet delivered model — and a complete revamp all of the instructional content in the product. Despite the best efforts of those involved, the new versions of SuccessMaker were failing to meet Pearson's internal goals and the needs of its most important customers. The difficulty of the situation was painfully illustrated in an unsolicited letter from a strategic customer delivered early in Adam-Woodford's tenure. In this letter, the customer bitterly complained of broken promises and missed expectations since work on the new version had begun 18 months earlier. This letter ultimately became the rallying cry for a complete reengineering of the SuccessMaker product development process. Adams-Woodford and his team undertook a complete flash-cut[1] from a more traditional and established development process (waterfall) to a new (and unproven within Pearson) development methodology (Agile) (see Exhibits 1 and 2). Agile's focus on short development iterations and communication over documentation was fundamentally different from the waterfall approach they had previously used, which emphasized complete and detailed documentation before development began. Now, a year later, many of the previous challenges were behind them. Both Adams-Woodford's key customers and internal groups agreed SuccessMaker had gone through a completely positive transformation. Adams-Woodford was pleased that his clear vision for the future had overcome many of the most significant issues with the development of SuccessMaker.

[1] *"Flash-cut" is a standard term for the immediate implementation of a new methodology or system.*

The new path forward was not as clear, however. The development team began to experience growing pains with the specific Agile methodology they chose to implement — Scrum. Some influential members of the team proposed Scrum should be abandoned in favor of a Kanban system. Kanban was developed in the manufacturing industry and was more focused on individual task management whereas Scrum focused short development cycles called sprints that included a whole set of tasks (see Exhibit 3). These trusted team members felt the Scrum methodology did not fit with the current lifecycle phase of the product. The data seemed to support this conclusion as the team's initial increases in productivity were beginning to trend back down. While Kanban was clearly successful in manufacturing firms, Adams-Woodford had no personal experience using it as a software development methodology. His initial reaction was to stick with what he knew. Adams-Woodford had advocated continually for the Scrum methodology over the past year; so the thought of throwing those processes away and implementing a new system did not sit well with him.

There was also the question of whether to extend Agile into other teams involved in developing SuccessMaker. In his strategic planning meetings the previous week, several of Adams-Woodford's peers suggested he take a broader role inside Pearson, assisting other development groups with implementing the Agile system. While the new responsibility was appealing, Adams-Woodford worried these new challenges might jeopardize his focus during a critical time for the broader SuccessMaker team as they evaluated transitioning to a new Agile system beyond software development alone.

PEARSON AND PEARSON DIGITAL LEARNING

Listed on both the London and New York stock exchange (LSE: PSON; NYSE: PSO), Pearson had three primary business units. The Pearson Education unit was focused on the education market and was one of the leading companies that developed learning products for teachers and students. The Financial Times group provided business and financial news publication and online services. The Penguin group published books in a wide range of genres under the Penguin brand.

Pearson reported $US 8.7 billion in revenues in 2010 (see Exhibit 4). Pearson Education group was the largest of the three divisions and accounted for 80 per cent of the firm's profits. The North American education market was the largest business segment for Pearson and accounted for 46 per cent of the firm's worldwide revenues.[2] The business sustained its growth even during the deep recession of 2007-2008. Pearson's myLab — a digital learning, homework and assessment service — grew to over 7.3 million registered students in 2010. A recently introduced suite of learning management technologies aimed at the higher education market — eCollege and Fronter — also had online student enrollment of over 8.3 million. Pearson Digital Learning focused on digital supplemental products, viewed as a market segment with strong potential for growth.

Despite an overall strong growth record, Pearson Education faced some headwinds as it moved into 2011. Continuing slow economic growth put considerable strain on education budgets in the United States. The budget crisis had a direct effect on education budgets at all levels. Pearson attempted to address these market difficulties with a focus on providing premium products and services. Additionally, they sought to take advantage of schools shifting to online and digital curriculum, instead of relying exclusively on the demand for traditional textbooks. Pearson also made some key acquisitions to keep offering a broad range of services to the education market. The firm recognized technological changes as the biggest risk to all its product lines and services. Due to these changes, transforming Pearson Education's products and services for digital channels was a key element of the firm's growth strategy.

[2] *www.pearson.com/investor/ar2010/performance/education/north-american-education.html accessed on January 12, 2012.*

SUCCESSMAKER

SuccessMaker was one of Pearson Digital Learning's (PDL) products focused on helping elementary and middle school students learn reading and math at their own pace. SuccessMaker was an all-digital product and was highly popular with the students who used it. The software also provided teachers with valuable assessment data, enabling targeted interventions and coaching. SuccessMaker was customized to individual markets to meet state mandated standards at each grade level.

The SuccessMaker product was derived from one of the earliest works of research on computer-aided learning. Researchers Patrick Suppes and Richard Atkinson lead the development of a decision theory based instructional program in the 1960s that eventually was marketed by the Computer Curriculum Corporation. In 1990, Simon & Schuster acquired Computer Curriculum Corporation. Pearson made a successful bid for Simon & Schuster's education operations (including the SuccessMaker product) and created the Pearson digital business unit in 1998.

The underlying approach to the design of SuccessMaker had remained unchanged since its creation. Students were given specific assignments to complete by the teacher in the SuccessMaker program (see Exhibit 5). Based on how students responded to these assignments, the software used a sophisticated, predictive algorithm to move the students across multiple skill levels of curriculum. Students who quickly mastered skills could move forward more quickly than students requiring additional assistance. The software was intelligent enough to know not only where to initially place students in a given set of curriculum, but also the optimal way to pace them through it. Through daily use of SuccessMaker, students were often able to achieve significantly better results on standardized tests compared to students using other instructional methodologies or products.

While SuccessMaker had undergone many improvements over the years, the most recent version was part of a major technology and content upgrade that began in 2005. This new version, codenamed *SuccessMaker: Next Generation* was intended to offer customers a more robust and scalable platform as well as a more modern curriculum. This new curriculum used a variety of new instructional tools including 3-D animation and an interactive avatar to guide the student through the program.

While the intent of the new release was to address critical market issues, the reality was that it was missing the mark. When Greg joined Pearson in 2007, he told the senior vice-president of product management, "We have spent $30 million on the new SuccessMaker development and even six to eight months after launching the first version, it has not been well adopted by the market. The few early adopters complained it did not meet any of their requirements."

One of Greg's first insights into the root cause of SuccessMaker's problems was that the development team was driving all the decisions instead of product management. As a result, there was little direct communication between the development team and customers causing many well-intended features to completely miss the mark when they were released. The SuccessMaker team used the waterfall development methodology (see Exhibit 1), emphasizing freezing product requirements at the beginning of a development cycle through formal documentation. With a major portion of software being developed offshore in India, miscommunication between the development teams in India and the United States was a common occurrence. Due to contractual constraints, changes to product features midway during the development cycle were either costly or impossible.

The waterfall process also required the organization to commit to a multi-year road map of features that did not allow for any significant changes in scope. These features were individually budgeted and committed in the business unit's strategic plan. Any changes to the road map were generally viewed negatively. The team referred to such changes using the pejorative industry term "scope creep." The waterfall process did make internal planning easier as the scope was essentially frozen for multiple years. It also meant little effort was put toward confirming committed features would meet customer needs or exploring alternative, unanticipated features once the road map was set.

THE PUSH FOR AGILE METHODOLOGY

Adams-Woodford successfully used Agile software methodologies at two early-stage firms prior to joining Pearson. Agile software development methodology advocated an iterative and evolving approach to software as opposed to the formal stage model of the waterfall approach. Agile methodology enforced a customer-driven approach to development with short development cycles — typically two week "sprints" delivering a small subset of product features using very little formal documentation. Short, "user stories" written on a single index card replaced the 100-plus page requirements documents of the past. There was no commitment to future sprints until the previous one was complete. This deferred commitment approach enabled teams to adapt, design and prioritize product features through ongoing feedback from product managers and customers.

A key assumption of the methodology was that product management *owned* the product, not the developers. Adams-Woodford lectured the development team on the role of a product manager, "Customers don't pay you to create documents; they want you to create a product. Product management must understand the market, distinctive competence, listen to its customers and prioritize features that create the highest value for its customers."

Adams-Woodford's first task was to change the basic structure and culture of the organization, starting with the software development team, including the developers, quality assurance (QA) engineers and product managers. The overall product development team was composed of a diverse group of individuals with expertise in different areas inside Pearson — product managers, software developers, QA experts, business analysts, curriculum/content experts, instructional designers, animators and artists. Before the formal shift to Agile, Adams-Woodford transitioned all offshore software development to in-house by hiring several new software developers in Arizona. Agile methodology required close cooperation between team members on almost a daily basis. Communication with the off-shore team had been a constant point of contention both with the Pearson team and their vendor. Adams-Woodford realized he had to create a more productive dialog focused on building a better product.

THE MOVE TO AGILE

The transition was executed as a flash-cut from the previous waterfall process to the new Agile methodology. On a Friday, the software development teams had been following the waterfall process and when they returned the following Monday, they were 100 per cent Agile. The move was timed right after the second major version of the product had been released to manufacturing (burning to DVD/CDs). This dramatic change met with skepticism from a vocal subset of team members. Adams-Woodford stressed the transition was as much a cultural shift as it was a change in processes. A few members quit as the changes were instated.

As soon as the cut was made, training was delivered to all the software developers, QA engineers and product managers, detailing the new methodology and everyone's role within it. Adams-Woodford brought in an independent Agile coach to teach the team about the new development methodology. All the teams spent two full days with the coach and the training was conducted in three phases. The first phase focused on explaining the underlying rationale of an Agile approach to ensure everyone understood the core Agile principles. The second phase addressed working within an Agile development framework, discussing specific work practices required. The final phase focused on the SuccessMaker team, asking members to discuss why Agile would not work in their environment. Proactively addressing these concerns was suggested by the Agile coach, who had previously observed lingering concerns about Agile often manifested in team behavior weeks or even months after making the shift from more traditional processes.

Since all teams and their top management were present in the training sessions, it resulted in the teams jointly exploring solutions to most of the problems that were surfaced. They spent a considerable amount of time discussing how some groups that were not going to be in Agile mode would engage with developers and product management groups given they were still integral to the larger development effort. Another issue that was discussed was that the development and QA teams had worked as separate and largely autonomous groups previously. It was agreed that the two teams had to be combined in an Agile environment, with extensive cross-training in their respective specialties. Under this new system, the most important change involved how product management would engage with development. Product management would now drive all development efforts and prioritization of new features. Product management would need to prioritize user stories based on market needs on an on-going basis — an uncomfortable change for both the development and product groups. Adams-Woodford and Eric Wagner — the newly hired vice-president of the development teams — tried to help the SuccessMaker team with this transition by ensuring they both were consistent and unwavering in their application of the Agile process.

The development teams were reorganized for SuccessMaker Release 3 as they finished their development-related work for Release 2. After some initial experimentation with different team sizes, the development team was split into four groups. Two teams each were responsible for development work involving the Java programming language. One team was responsible for handling coding of reports and curriculum logic. Another team created all the Adobe Flash-based animations required for the product. In addition to these teams, a *scrum master* was designated in each team to assist the team in removing any roadblocks.

GROWING PAINS FOR RELEASE 3

Adams-Woodford made a point to walk the halls and interact with as many of the team members as possible to keep in touch with their feelings and attitudes about the ongoing changes. After the initial novelty of the new methodology began to fade, Adams-Woodford observed development teams trying to reintroduce some of their previous waterfall processes. Specifically, he noticed the QA team was not comfortable with their new roles. As he chatted informally with members of this team, it became clear many of them were not sure how to go about their jobs without being able to rely on detailed requirements documents as they had in the past. This impacted the team's relationships with the developers, who felt individuals on QA team were not committed to the fundamental tenets of Agile methodology (see Exhibit 2) and were actively trying to return to previous practices.

The main difficulty facing the QA team was its joint responsibility with the developers for the quality of the release. This meant previously clear lines between what a developer did and what a QA engineer did had blurred. Previously, the QA engineers had no interaction with the developers and simply waited for the code to be completed so they could check it against the test cases developed independently from the original and very detailed requirements documents. If they discovered software defects, the engineers' only responsibility was to identify the issues and pass them back to the development team. In the new model, there were no requirements documents and the entire team — developers and QA engineers — was responsible for delivering working code at the end of the defined sprint. This team approach meant everyone needed to contribute with little regard for formal functional roles. Individuals on the QA team struggled with not only this ambiguity, but also with what they perceived to be a lack of accountability and trackability for features they were developing.

Adams-Woodford also noticed managers of both the developers and QA engineers were increasingly involved with task management on the sprint teams. The scrum process allowed managers to attend the daily stand-up meeting, but also dictated no one other than the members should be involved in team-level decision-making during these meetings. The meetings were called *stand-ups* given all attendees stand during the meeting in order to encourage the team to be concise and keep the meeting to no more than 15 minutes. Adams-Woodford attended all the stand-ups and found that he increasingly had to remind managers the teams were to be self-managing in relation to the daily plan. Adams-Woodford believed that unless the managers were completely committed to their employees managing the daily work, the benefits of Agile would be significantly reduced.

Of all the people he spoke to during his hallway conversations, the product managers and owners seemed to adjust most quickly and experienced the most dramatic change in their outlook for the future. In the scrum process, the product managers and product owners were the business representatives responsible for creating and prioritizing the user stories for each development sprint. Adams-Woodford recalled his early days at Pearson when these individuals were by far the most jaded and ambivalent of any of the cross-functional teams. They all seemed resigned to having little control over what the development teams produced. Most of their time was spent performing damage control when new releases failed to meet expectations. Now, however, these managers and owners were committed to the new methodology, becoming outspoken advocates of the Agile mindset across the business unit. Several took it upon themselves to take classes and join discussion groups about the methodology, while many established good relationships with both customers and the sales team.

When Adams-Woodford was not out talking to the teams, he was in his office reviewing key performance metrics for the development teams. Since the switch to Agile, the code base for SuccessMaker had been cut by half even while the number of features had increased by up to 75 per cent. The new features had been very favorably received by clients. The development team size had also shrunk by at least a third, mainly through eliminating off-shore development teams. Despite these improvements, the first few sprints of Release 3 showed significant performance degradation in the velocity of the developers. Velocity was calculated by converting each user story into a standard measure of effort as story points and determining how many points were completed in each sprint. Adams-Woodford did not think it was a coincidence when the velocity began to slow as the teams began regressing into their previous waterfall processes. This development prompted Adams-Woodford's key developers to suggest a change from Scrum to Kanban in order to jumpstart the team's overall velocity.

THE PATH FORWARD

While the move to Agile development improved SuccessMaker's position with both customers and the sales team, a number of challenges remained outside the software development team. Out of several different cross-functional teams, the software development team was the only functional group to transition to Agile. Other teams involved with the product such as the content team still used a waterfall approach. Team leaders on the software team also recognized a number of issues requiring immediate attention beyond the QA engineers' concerns about their roles and software testing. End-to-end integration testing and complete automation of the testing procedures were still works-in-progress. There was a tendency to follow a *mini waterfall* approach when this testing was required. The teams wanted to avoid this tendency in future.

As far as planning for the final release, which was made up of many sprints, the product owners were pleased to note they were able to plan up to two sprints for each development team accurately. As all the teams achieved higher levels of maturity in Agile methodology, they hoped to plan out to three to four iterations. This required greater stability and predictability in achieving the iteration goals. Adams-Woodford realized this weakness was an important hurdle to overcome, stating, "Teams don't consistently understand what their velocity is. They still cannot estimate accurately how many story points they can develop in a planning period. This makes it hard to commit to releases with customers." Adams-Woodford also worried teams were becoming too internally focused on estimating a static backlog of features. Unless they were also continuously validating that the requirements in the backlog were the most important to the market, these teams could begin to slip back into developing features unable to meet the market need.

Beyond the planning of the sprints, the activities within an individual sprint had room for improvement. David Foster, the chief architect for the division, noted the product development teams engaged in *context switching*. In other words, development teams often switched from one feature to another distinct feature within the same development iteration. This resulted in reduced velocity since the developers had to review and familiarize themselves with technical aspects of the system components each time they switched from one feature to another. To eliminate context switching, Foster noted the need for the product and development teams to recognize the mapping between features and systems components. Foster also observed, "We haven't become very efficient at maintenance releases. With Scrum, it shouldn't matter, but we are still not comfortable with them. That is still a rough spot."

Adams-Woodford often insisted the developers adopt a startup mentality, stating, "We are diversifying our product lines rapidly. We are going to have to be creative in converting something that some people might think is really boring into something very compelling and focus on our distinctive competence." For this to happen he noted that he needed to keep his development teams fresh and energized. Agile development frameworks can make software development a "grind" and monotonous. Thus, it was important for him to keep his developers engaged with the products and customers. To keep them inspired and motivated, Adams-Woodford encouraged cross-team deployment of development team members across the product family.

Adams-Woodford also saw challenges in the product's strategic position in the market, stating,

> "When you are using SuccessMaker, it still feels like a dated product. We have made great strides in the platform technology, but our content road map has remained essentially the same for the past three years. The product is also expensive to maintain compared to other web-based, next-generation products. We have to provide different options — hosting, subscription-based service etc. There are only so many features you can add to the product. The strategy is to take the good brand, build ancillary products, reach into the home and connect with the parents and students. We have to make the product entertaining for kids to play on their own at home, while it reports back to the teachers and parents. We must use the data, make it interoperable at the district level. Inside Pearson, we must make sure that the data is portable across products — this is really important."

All of these issues swirled around Adams-Woodford’s mind as he considered how to approach the new product road map and where he would focus his efforts in the future. Although Adams-Woodford had made great progress inside Pearson by implementing Agile with the development teams, the challenges he faced now were much more complex and potentially more important in the long-run for the prospects of his product portfolio.

“Well,” Adams-Woodford thought to himself. “If Agile works for development, maybe it will work for me.” Adams-Woodford went to the developers’ planning area and brought back a pack of index cards to his office. Adams-Woodford began to outline user stories for how he would implement the product road map, his place within it and the broader Agile effort at Pearson.

Exhibit 1

A SUMMARY OF WATERFALL SOFTWARE DEVELOPMENT APPROACH

Software teams have traditionally followed a phased approach when building systems. The term "waterfall" refers to the inherent rigidity in development lifecycle where one phase is presumed to begin only after the completion of a previous phase. However, it is common to see software teams iterate between phases. The main phases in the software development lifecycle (SDLC) include — (1) requirements analysis, (2) design, (3) development, (4) testing, (5) deployment and (6) maintenance. Managers emphasize the need to extensively analyze and document the user requirements since they consider changes made to requirements and design during the latter stages of the development lifecycle to be highly expensive and disruptive. As a result, phased development models are documentation intensive.

A key assumption of a phased approach is that the dependency between phases is linear and uni-directional. However, in most software projects, dependencies between components are complex and bi-directional. To address this complexity, managers attempt to split the projects into relatively independent modules and integrate the modules towards the end of the project lifecycle.

Despite adaptations to the basic concept of a "waterfall" model, this development philosophy has many drawbacks. In most cases, it is difficult for users to commit to requirements since they have difficulties in visualizing the final product. Moreover, business requirements may change during the course of development and cause the project to backtrack to earlier phases thus leading to wasted efforts. Unrealized dependencies between systems, components or modules may lead to long delays during systems integration. As a result, many software development projects have embraced iterative development approaches such as Spiral, Rapid Application Development or Agile models.

Source: Richard E. Fairley, Managing and Leading Software Projects, John Wiley, Hoboken, New Jersey, 2009, pp. 55-58.

Exhibit 2

AGILE SOFTWARE DEVELOPMENT METHODOLOGY

To understand the philosophy of Agile development methodologies, it is helpful to note the twelve principles as proposed by the Agile Alliance:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity — the art of maximizing the amount of work not done — is essential.
- The best architectures, requirements and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile methodology takes an adaptive approach to development. At the end of an iteration (known as *sprint*), the team delivers a set of specific features that are tested and integrated into the product. Teams have the opportunity to review progress and revisit user requirements at the beginning of each iteration. As a result, all stakeholders have the opportunity to react to changing product needs and provide inputs on evolving product features.

One of the more popular variants of an Agile process is the Scrum approach. In a scrum team, the scrum master represents the developers and manages the development process. A product owner prioritizes features based on stakeholder needs; and the development teams directly engage the product owners and other relevant stakeholders continuously throughout the development process. Agreement on what features to develop in a sprint cycle is made during a planning meeting. In a sprint cycle, the development team strives to meet the feature commitments made during the planning meeting. Teams and stakeholders typically meet at the end of the sprint cycle to review progress and discover opportunities for continuous improvement.

Source: www.agilealliance.org/the-alliance/the-agile-manifesto/the-twelve-principles-of-agile-software/ accessed January 12, 2012.

Exhibit 3

KANBAN

Kanban is a Japanese term (看板) which literally translates to “signboard.” Kanban as a production process has its roots in Lean manufacturing and the Toyota Production System (TPS). Kanban focuses on optimizing the flow of the work through the entire production process by managing the requests for additional components/work from upstream functions. These requests are made via kanban cards that downstream functions pass to upstream functions when more components are required. The cards are visible throughout the production process, so it is possible to literally see the flow of work by examining all of the cards and their queues. Kanban is one tool that is commonly used in Just in Time (JIT) manufacturing given additional inventory is only requested as it is needed. Kanban is also known as a “pull” system given downstream processes pull additional work from upstream functions through the use of these cards. Beyond the use of cards, a Kanban system includes the analysis and optimization of the production flow in order to decrease the waste (known as muda or 無駄) by imposing limits to the amount of work that is in process within the system and individual functions. This optimization seeks to improve the overall throughput of the production process.

Kanban as a software development methodology is a more recent adaptation of the traditional manufacturing application. Kanban in this context is often associated with Lean software development processes. In software, Kanban embraces the philosophy of the manufacturing implementation and applies it to various steps of developing a product or application. These steps often include user story creation, development, test and deployment. Unlike more traditional Agile processes like Scrum, Kanban does not utilize a *time box* for each development sprint. Instead, Kanban imposes work in progress limits for each development step. The overall process can be optimized by imposing and adjusting these limits. Team members focus on system bottlenecks to reduce the overall feature throughput time by eliminating waste (muda) throughout the process. Unlike other Agile processes which use velocity as their key metric, Kanban uses throughput for individual features as its key metric. Kanban has some similarities with more traditional Agile processes including allowing business owners to put off feature commitments until the feature enters development and seeking to allow teams to work together more collaboratively by making the workflow, which can be physical or components of a software application, visible on the Kanban board.

Source: D. Anderson, Kanban: Successful Evolutionary Change for Your Technology Business, Blue Hole Press, Sequim, 2010, pp. 11-16.

Exhibit 4

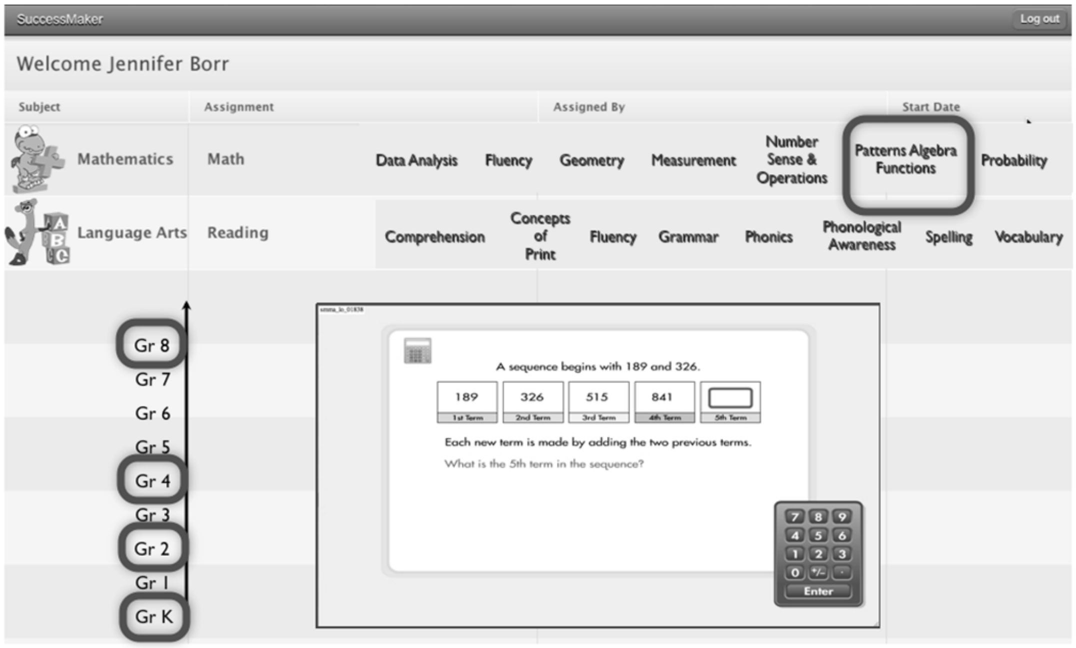
PEARSON FIVE YEAR FINANCIAL PERFORMANCE FROM 2006-2010
(IN $ MILLIONS EXCEPT PER SHARE)

	2010	2009	2008	2007	2006
Sales	8,716.49	9,092.32	7,033.20	8,258.66	8,102.73
Cost of Goods Sold	3,715.63	3,847.75	2,991.05	3,625.32	3,619.49
Selling, General, and Administrative Expense	3,544.78	3,697.39	2,803.92	3,266.16	3,221.90
Operating Income Before Depreciation	1,456.08	1,547.18	1,238.23	1,367.18	1,261.34
Depreciation and Amortization	358.63	375.08	286.53	273.83	250.70
Interest Expense	126.21	148.74	154.96	226.21	229.16
Nonoperating Income (Expense)	76.96	43.65	58.48	61.51	131.23
Pretax Income	1,031.26	1,067.02	855.21	928.65	912.71
Income Taxes - Total	224.72	320.11	251.45	259.94	21.55
Income Before Extraordinary Items	814.24	687.10	558.45	617.12	846.12
Extraordinary Items and Discontinued Operations	1,182.11	0.00	(131.57)	(53.58)	27.42
Net Income (Loss)	1,996.34	687.10	426.88	563.54	873.54
Earnings Per Share (Primary) - Excluding Extraordinary Items	1.02	0.86	0.70	0.77	1.06
Earnings Per Share (Primary) - Including Extraordinary Items	2.49	0.86	0.54	0.71	1.10
Common Shares Used to Calculate Primary EPS	801.20	799.30	797.00	796.80	798.40

Source: Standard & Poor's Computstat Data, accessed November 12, 2011.

Exhibit 5

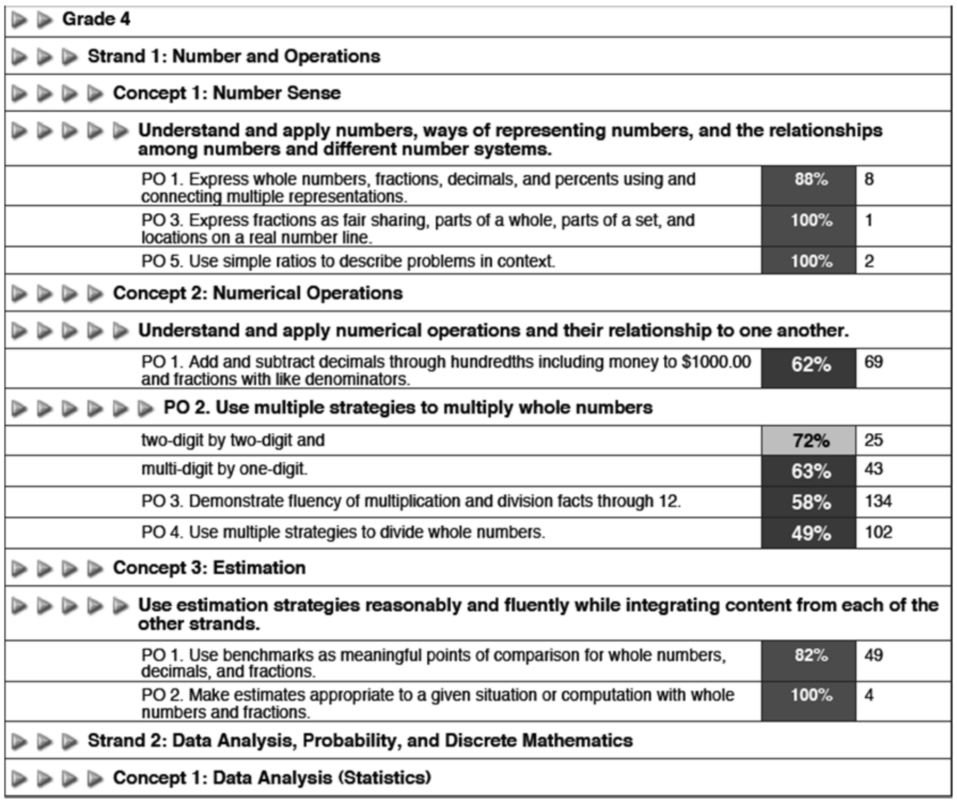
SUCCESSMAKER



A sequence begins with 189 and 326.

Each new term is made by adding the two previous terms.

What is the 5th term in the sequence?



▷▷ **Grade 4**		
▷▷▷ **Strand 1: Number and Operations**		
▷▷▷▷ **Concept 1: Number Sense**		
▷▷▷▷▷ **Understand and apply numbers, ways of representing numbers, and the relationships among numbers and different number systems.**		
PO 1. Express whole numbers, fractions, decimals, and percents using and connecting multiple representations.	88%	8
PO 3. Express fractions as fair sharing, parts of a whole, parts of a set, and locations on a real number line.	100%	1
PO 5. Use simple ratios to describe problems in context.	100%	2
▷▷▷▷ **Concept 2: Numerical Operations**		
▷▷▷▷▷ **Understand and apply numerical operations and their relationship to one another.**		
PO 1. Add and subtract decimals through hundredths including money to $1000.00 and fractions with like denominators.	62%	69
▷▷▷▷▷▷ **PO 2. Use multiple strategies to multiply whole numbers**		
two-digit by two-digit and	72%	25
multi-digit by one-digit.	63%	43
PO 3. Demonstrate fluency of multiplication and division facts through 12.	58%	134
PO 4. Use multiple strategies to divide whole numbers.	49%	102
▷▷▷▷ **Concept 3: Estimation**		
▷▷▷▷▷ **Use estimation strategies reasonably and fluently while integrating content from each of the other strands.**		
PO 1. Use benchmarks as meaningful points of comparison for whole numbers, decimals, and fractions.	82%	49
PO 2. Make estimates appropriate to a given situation or computation with whole numbers and fractions.	100%	4
▷▷▷ **Strand 2: Data Analysis, Probability, and Discrete Mathematics**		
▷▷▷▷ **Concept 1: Data Analysis (Statistics)**		

Source: Pearson Digital Learning.