IMD-7-2152
21.11.2019

SOFTPLAN UNIC: BECOMING AGILE (A)

Research Associates Sophia Shilimindri and Rodney Reis prepared this case under the supervision of Professor Carlos Cordon as a basis for class discussion rather than to illustrate either effective or ineffective handling of a business situation.

MAY 2017. Guilherme Brasil had just taken over as chief technology officer of UNIC, a division of software company Softplan developing and selling enterprise resource planning (ERP) software for the civil construction industry. It was a difficult time in UNIC's history; the unit's growth had stalled while costs continued to rise and profitability was expected to vanish in less than five years. Following a request from the board to the company's management for a growth acceleration plan, Guilherme had been working with Ionan Fernandes, UNIC's COO, for the past nine months on an ambitious project to modernize the company and drive growth. UNIC's long-term survival depended on this plan.

Concerned about the impact of the changes the plan might have on short-term profitability and employee retention, the board was following its progress closely. UNIC's growth strategy was centered around changing its business model from selling on-premises ERP software licenses to providing its software as a service (SaaS) in recurring contracts. This strategy could only be delivered by creating the right infrastructure and having the right people to implement it. At that point, it was clear that UNIC lacked both.

Guilherme had three months to present a plan that would enable this business model transformation. From his experience and from examples of what other companies had done, he felt he had to choose between one or a combination of the following options:

1. Acquire a SaaS company and grow based on its products, people and culture.
2. Hire a workforce skilled in SaaS technologies and agile practices to replace several of the current team members who lacked these skills.
3. Transform the unit's culture from within by educating the workforce.

It was not a straightforward decision.

Softplan (UNIC's Parent Company)

Softplan was one of the largest software companies in Brazil, offering corporate software solutions for justice, public administration and construction markets, among others. Located in Florianópolis, it had been steadily growing since its incorporation in 1990 and by 2018 it had over 1,700 staff and more than 3,500 clients in Brazil and internationally.

Softplan was comprised of three business units with different target markets all operating independently while sharing corporate functions (HR, IT, Finance). Three partners privately owned the company, each with full management control over one business unit, making sure that it was profitable and sustainable on its own. Two of the business units offered software solutions for the public sector, while the third, UNIC, was one of the leading ERP solution providers in Brazil. Its product, Sienge, was used by most middle-sized construction companies in the country as well as by some of the largest ones. It had been developed by the three founders some 30 years earlier to fill the need for ERP in the construction industry.

In 2016, following a series of government corruption scandals, the Brazilian economy shrank by 3.6%, and the country saw its worst recession in 25 years. The construction industry was one of the worst affected with thousands of jobs lost. Seeing its clients in crisis, UNIC's management had reason to be concerned. Austere times meant cost reductions and the risk was even higher in the construction industry. With less than 1% of revenues spent on IT, it was known for being one of the least digitized sectors. UNIC needed to make drastic changes to survive, starting with its business model. Projections showed that if the current trends around revenue and operational costs continued, it would go from being profitable to needing additional financing in less than five years.

Failing to Innovate

The software industry had seen immense growth and innovation during the previous decade, fueled by technological advancements, new business models and enormous capital investments. The fact that UNIC's innovation efforts over the last five years were not delivering the desired results was extremely worrying for its management. While the company was growing, the current pace of innovation was not enough to sustain UNIC's market leader position.The commercial success of its ERP product would be short-lived if it did not continue to innovate its business models, products and technologies. Failure to do so could potentially cost the future of UNIC.

Many of new product launches over the previous five years had failed for various reasons. At this point, UNIC had 38 innovation projects running at the same time as well as 24 corporate change projects. However, nothing significant had been produced out of any of these projects. Guilherme saw the failure to innovate as a collective failure of every single UNIC employee:

> Innovation should not come from a specific area, but from the company as a whole, fostered by a group that thinks about new business perspectives.

UNIC's employees needed to learn how to innovate, and the first step of this learning curve would come from the management in the form of business plan innovation.

Software as a Service

UNIC's business model in 2016 relied mostly on selling enterprise software with the perpetual license model making up 80% of its revenue. The other 20% of its clients were buying its software as a service (SaaS) on the cloud, paying an annual fee.

UNIC felt that it was both risky and unsustainable to continue basing its business on perpetual licenses. Sales were highly susceptible to changes in the market and the economic environment. The one-off

investment in Sienge might bring efficiency benefits to clients in the long term but in times of financial difficulties, it was one of the first items to be taken out of the budget.

The entire software market was moving toward SaaS, and new start-ups were continually coming to market offering much less complicated and lower-cost products that would eventually start impacting UNIC's core market in a classic case of the "innovator's dilemma." UNIC's clients were slowly getting used to buying software on a subscription basis. Many preferred to have it as a small recurring cost instead of a significant upfront investment, and not to have to invest and maintain servers to run their systems. This trend was not only beneficial for the users but also for the vendors. Ultimately, the average lifetime value of a client that bought a traditional license was lower than the average lifetime value of a SaaS client, but the upfront value was higher, impacting the short-term cash flow.

Table 1: Comparison of perpetual and SaaS license models

	Perpetual license model	**Cloud/SaaS**
Payment	One-off license payment upfront, possible recurring support fees and software upgrade* charges	Recurring payments for the license, support, upgrades and cloud-based operations
Software location	Mainly on the client's premises or on a private cloud	On the vendor's cloud
Software updates*	Long cycles (e.g. every quarter) – the client is often responsible for installation and maintaining the underlying IT infrastructure	Fast update cycles, could be multiple updates* per day – vendor is responsible for installation and cloud infrastructure

*Upgrades include significant changes (e.g. technological modernization, additional functionalities) while updates refer to small changes to the software (e.g. bug fixes, adjustments in usability) and are usually covered by technical support fees.

In view of that evolution, it made sense for UNIC to shift its business model to SaaS, but to be able to achieve that change, it needed to build the capabilities to operate it. While it was acceptable for traditional license clients to wait months for a software update, typical SaaS clients expected a much faster service. Guilherme seriously doubted that the current organization could cope with having to offer clients this level of speed and experience.

With the perpetual license model, UNIC often had to customize its product according to specific client needs, increasing the complexity and size of the software code, resulting in significantly more time, effort and costs to fix bugs and deliver new features. With the move to a cloud-based SaaS version of Sienge, Guilherme saw the opportunity to mitigate the impact of client customization requests by providing direct access to the system modules through standard application programming interfaces (APIs), enabling distributors to develop applications on top of Sienge.

Opening Sienge interfaces to external developers would ensure a source of revenue for UNIC's distributors and at the same time eliminate the need to customize software for every client. However, by opening the system, UNIC would need to make significant changes in the software code, improve its documentation and modernize its web engineering implementation. Implementing these changes was highly dependent on the skills of the personnel and the organizational structure of the company. Guilherme believed that if he could get the corporate structure right, both the employees and the product would develop as a result. Two years before Guilherme became CTO, UNIC had changed the organizational structure to be more "agile." This change by itself, was not enough to bring the desired results. .

Becoming Agile

In 2001, a group of 17 software developers met at a resort in Snowbird, Utah to discuss their views and frustrations regarding the traditional project management methods used in software development. Together, they wrote and published the values and principles they believed should underline modern software development – The Agile Manifesto. Many successful and innovative companies around the world adopted these values and beliefs; UNIC was one of them. Agile management was adopted by all types of organizations and practiced in many aspects of work from operations to marketing.

The manifesto also raised essential questions to business leaders and management, for example, could focusing on customer collaboration divert the attention of employees from making a profit for shareholders? How can staff remain productive by prioritizing responding to change over following a well-defined plan? And most of all, how does an organization measure the success of the implementation of agile?

Despite criticism, agile management brought benefits to many companies. One example was Spotify, which became successful by having small independent teams quickly testing new features and learning from the tests instead of following detailed plans and requirement specifications. Spotify managed to surpass its competition by implementing an organizational structure that embraced agility.

The term agile was used to describe not only an approach to software development but also a specific way to organize a company into small, self-managed teams. Jeff Bezos from Amazon coined the "two-pizza rule" – if a team can't be fed with two pizzas, it is too big. In traditional companies, the staff was organized into departments according to functions. In a software company, one would typically find development teams, quality assurance and IT ops separated and managed independently. However, companies adopting agile could become even faster by organizing teams of people with different skills and specialties so that the entire product development lifecycle could be managed by one team in a methodology known as DevOps.

As a first attempt to implement agile methodologies and transform its operations, in 2016 Softplan organized UNIC's personnel into 12 self-directed teams of 8 to 12 people each. Teams were allowed to define their way of work, and as a result, even though they all started based on well-known agile methodologies, each team had set their own rules. Nobody questioned the effectiveness of this organizational structure. Senior management trusted their engineers since they were very experienced in their jobs. They felt that allowing the engineers to self-regulate would make the company more innovative.

Two years after the implementation of the new and more agile structure, the company was still not innovating as much as expected. Something had to give. When Guilherme took over as CTO, he feared that the company would not be able to transform itself from the inside, so inevitably, he started looking for solutions outside UNIC.

Acquiring Another Company

The option of acquiring a smaller and more innovative company was very appealing for UNIC. Many companies used mergers and acquisitions (M&A) to add new products and services to their portfolio, as well as gain new talents and skills. The plan was to buy a start-up with an already developed product for the construction industry and incorporate it as a separate business unit.

The opportunity came in the form of a local start-up, Target A, which had developed a SaaS product for the construction industry. From the outside, it looked like an ideal acquisition. It had a small team working based on agile principles and a product with a cloud-centric architecture just like the one Guilherme wanted to develop. The plan was to use it as a basis to build a complete cloud-based SaaS product and sell it to UNIC's clientele.

As the acquisition process advanced into the due diligence phase, Guilherme decided to hire a specialist company to gain a holistic view of the risks and opportunities associated with the organization's software assets and capabilities. The firm he chose was Avalia Systems, a Swiss company specialized in software due diligence for investments and acquisitions, to make sure he had all the necessary information before buying Target A. Avalia had a unique approach to performing software due diligence (*refer to **Exhibit 1***). Its team of business and software experts focused on the business impact of technology and used data from software development systems to discover insights and define actions to reduce risk and improve business performance post-investment.

Avalia had developed a unique software analytics system in collaboration with the Swiss University of Applied Sciences HEIG-VD. The system collected data from tools used to develop software, analyzed it and used the results to drive discussions with key people in the target. More specifically, the analytics system used data from the systems, such as source code version control, issue tracking and code quality analysis as well as interviews and surveys with the software development team. The resulting insights were often surprising and provided the basis for a detailed action plan to secure the investment deal.

The due diligence team:

- Analyzed the data of the software used by the team to do their work
- Reviewed the system's technical architecture and critical areas of its software code
- Conducted interviews with the teams to understand their way of working
- Created a survey to gather data from employees about how they worked.

Based on the data, interviews and survey, the team analyzed who developed the code, how the team collaborated, the quality of the code, how much time it took to build new code vs. repairing problems with the existing code, the complexity of the code, the timing for new releases, and other indicators to create a holistic view of the performance of the company as an agile software enterprise.

One of the most revealing findings came from the analysis of the code lines each developer had created (*refer to **Exhibits 2 & 3***). The first surprise was the number of people contributing to the code. The expectation was to find a team of six developers, while the data showed that in the previous year only three people had made significant contributions to the code. When asked, Target A's CEO explained that one developer had left the company more than a year before. He also said that one of the developers had to move to pre-sales to support client rollouts and that one of the team members was in fact a product owner, who rarely coded but was nevertheless part of the development team.

Another interesting result from Avalia's analysis was that only 5% of the software code had changed in the past year (*refer to **Exhibit 4***).

Hiring New Blood

In addition to ensuring that it acquired a suitable target company, UNIC also had to decide how it was going to integrate the target into its current structure. One thing was certain, if it was going to develop the Target A product, it needed to have additional resources working on it. Guilherme was toying with the idea of hiring new engineers since UNIC's current engineers were not up to date on the more modern technologies. Whether it acquired another company or not, it would eventually need people who knew the latest programming techniques to create a robust SaaS solution.

While hiring new and qualified engineers seemed like a good idea, Guilherme was not sure if he could find enough skilled engineers in the area. He therefore decided to test the concept by opening new positions. He hired a small team of experienced engineers to whom he assigned "advanced" projects.

However, given that his budget was limited, to bring new people into the company, Guilherme needed to let go a similar number of existing employees. Letting go of an employee was not a decision Guilherme would take lightly and the Softplan's owners, being engineers themselves and valuing their employees, disagreed with the idea of "restructuring" and felt strongly about firing the people that had helped them build Softplan into a market leader. Before presenting the idea to the board, Guilherme needed to have a thorough understanding of how his engineers operated. Having seen the in-depth analysis that Avalia performed on the acquisition target, Guilherme decided to engage the company to conduct the same analysis for his own software team:

> Once I saw Avalia's due diligence on an M&A deal, I knew they could help bring our software development team to the next level.

Unravelling the Spaghetti Bowl

Avalia used the same method of analysis as for the due diligence, starting by looking at the quality of the code. The first finding was rather worrying: the engineers were using an outdated source code management system. Having proper source code management was essential, not only for the quality of the product but also for the efficiency of the team. Without it, engineers could unintentionally overwrite changes that a colleague had made, leading to lost time, poor quality and frustration. When they were asked why they hadn't changed their system, the response was that they didn't have time to learn the new one.

The analysis of the different versions of the software code showed that the quality of the code was not improving over time. On the contrary, it was becoming more complex and intertwined with each revision (*refer to* ***Exhibit 5***).

A more thorough analysis of the software source code revealed valuable information on how the engineers worked. At that time, most of the Sienge software code was concentrated in a single big block. As a result, changing one piece of software could affect other functionalities. This way of coding made it nearly impossible to predict how changing a part of the software might affect other parts. It was like unravelling a big bowl of spaghetti. Every time something needed to be changed, software developers had to spend a significant amount of time debugging the software, i.e. fixing problems in the code, and every change resulted in problems (bugs) in another section (*refer to* ***Exhibit 6***).

Complexity was a classic problem of legacy software systems. The focus was always on delivering the changes as quickly as possible; downplaying how the structure of the system would affect future developments. Newer software used microservices architecture whereby the software was broken into small sections that passed information to each other through open interfaces like an assembly line. A useful metaphor is that of a car built based on a modular system, where the interfaces of the modules are well defined, and a module can change without affecting the others. Microservices enable third parties to take over custom client needs and monetize their development and support. For Sienge, changing the architecture to microservices signified that the software would go through a significant overhaul. It also meant that software development resources needed to be released to upgrade the software.

Changing the software ultimately was a necessity considering that the engineers spent a significant amount of their time dealing with "technical debt," an accumulation of a series of low-quality code which often resulted in "bugs" in the software. When a client reported a bug or a problem, it was registered in a database. The teams prioritized the work according to their workload and the urgency of the problem. Consequently, clients had to wait several months to fix a bug, and the teams spent most of their time fixing mistakes instead of developing new products (*refer to* ***Exhibit 7***). The problem was not that the personnel was not interested in improving. On the contrary, when asked, developers were well aware of the quality issues. Avalia found that the developers did not lack motivation as one would expect from people who have been comfortable working in a job in the same company for many years. They wanted to improve, but they didn't know how to start.

Trying to become more agile, UNIC had lost the ability to coordinate decisions across all its teams. This meant having to support a large number of clients and maintaining a huge legacy system while operating as a group of small companies. On one hand the development team valued the independence that agile practices had brought them, on the other they were apprehensive about making decisions such as adopting new tools, processes, and technologies that would impact the entire organization. Migrating their customers and partners to a new technological paradigm meant that they would have to make these decisions, get everyone onboard, and learn new ways to work. How could they do it without going back to a command-and-control philosophy that dictated decisions from the top?

The Road Ahead

Guilherme and Softplan's senior management understood that a lot of change was required to reach their objectives. The company's plan was to increase revenues by fostering innovation. It needed to be able to develop products with a short time-to-market. The first step was to create a multi-tenant SaaS platform and the second was to create an ecosystem by opening its system through APIs to third parties.

The second step was to reduce costs to be able to deliver value at a sustainable pace. Reducing costs meant improving productivity, in other words, making sure that the product quality was high enough that the staff spent less time fixing issues with the code (bugs).

The management and especially Guilherme had a difficult decision to take. Looking at what other companies had done, they had the following options:

Option 1: Acquire Target A and invest in growing its products, people and culture to gradually replace the legacy solution.

Option 2: Transform the current structure by dismissing a significant percentage of the existing workforce and replacing them with people that already had experience with the technologies and processes that needed to be implemented.

Option 3: Transform the company from within by educating the current workforce, changing the culture, and enabling them to do the work that needed to be done.

Questions

1. What type of changes does UNIC need to implement to transition its products to SaaS?

2. What are the advantages and disadvantages of having agile teams?

3. Considering the changes that need to be implemented, which of the above options would you choose, and why?

4. Aside from financial results, what other metrics would you use to measure the success of this change?

5. Do you think that UNIC should continue to embrace agile principles or would a more hierarchical structure work better?

Exhibit 1
Avalia's Data-driven Due Diligence

Source:

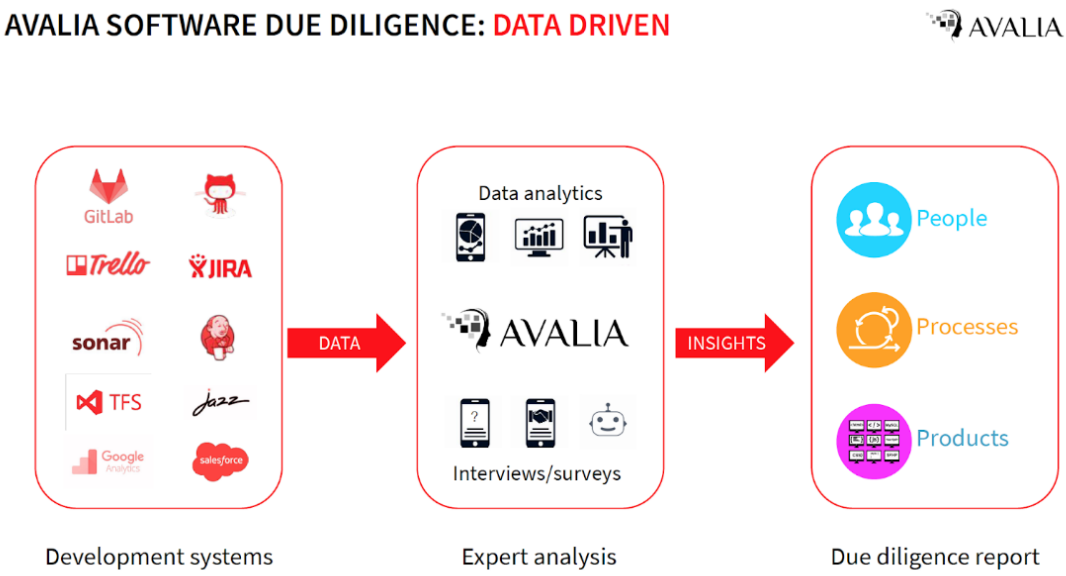


Exhibit 2
How Target A Employees Spent their Time the Previous Year

Source:

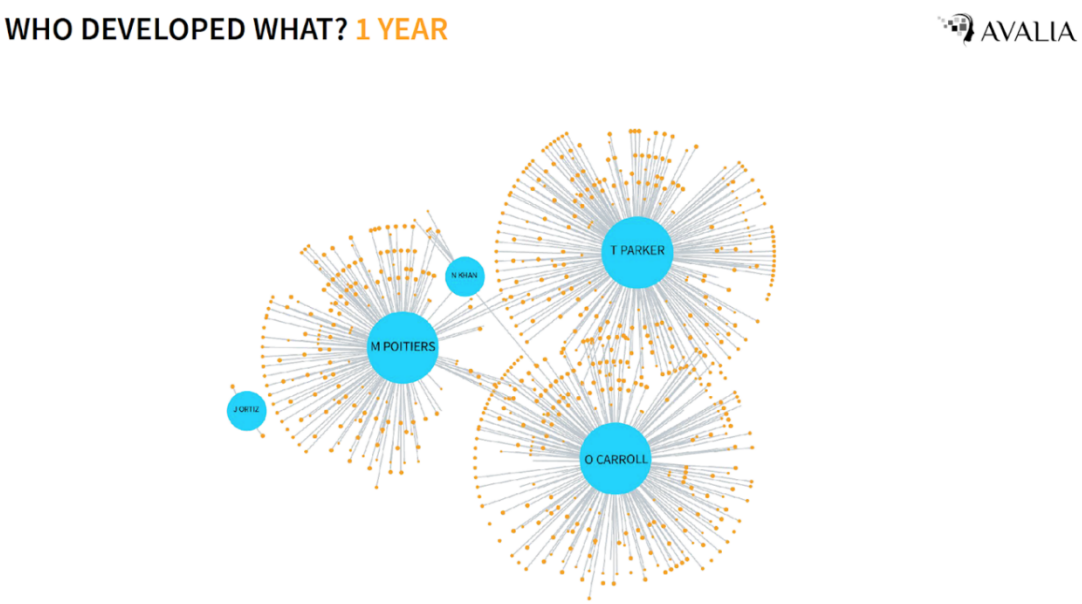
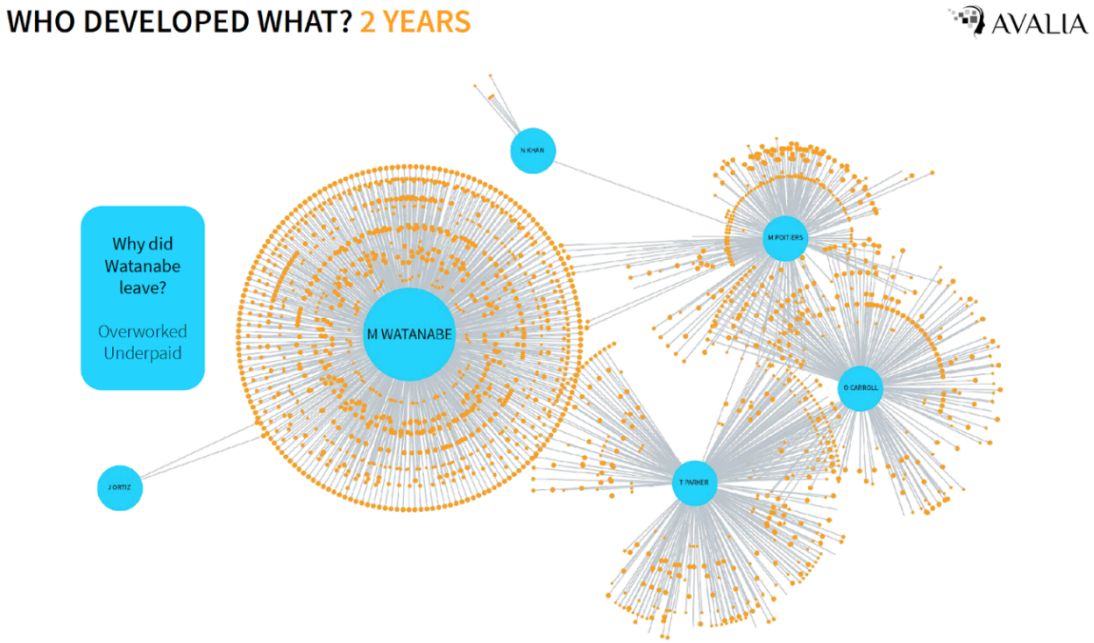


Exhibit 3
How Target A Employees Spent their Time the Previous Two Years



Source:

Note: Each blue bubble represents an employee, and each orange dot represents a code file. The size of the blue bubble represents the relative importance of a person's contribution to the codebase. The lines show which software developer(s) worked on which files, and how closely they collaborated.

Exhibit 4
Percentage of the Code of Target A that was Changed in the Previous Year

Source: Avalia Systems

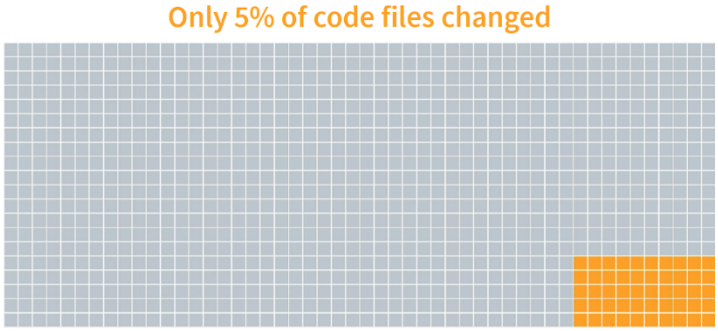
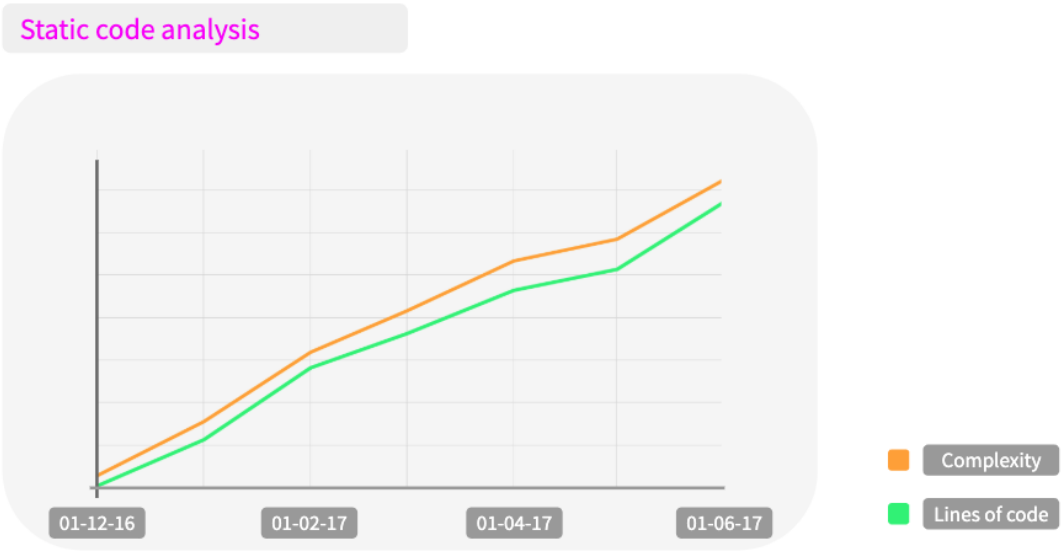
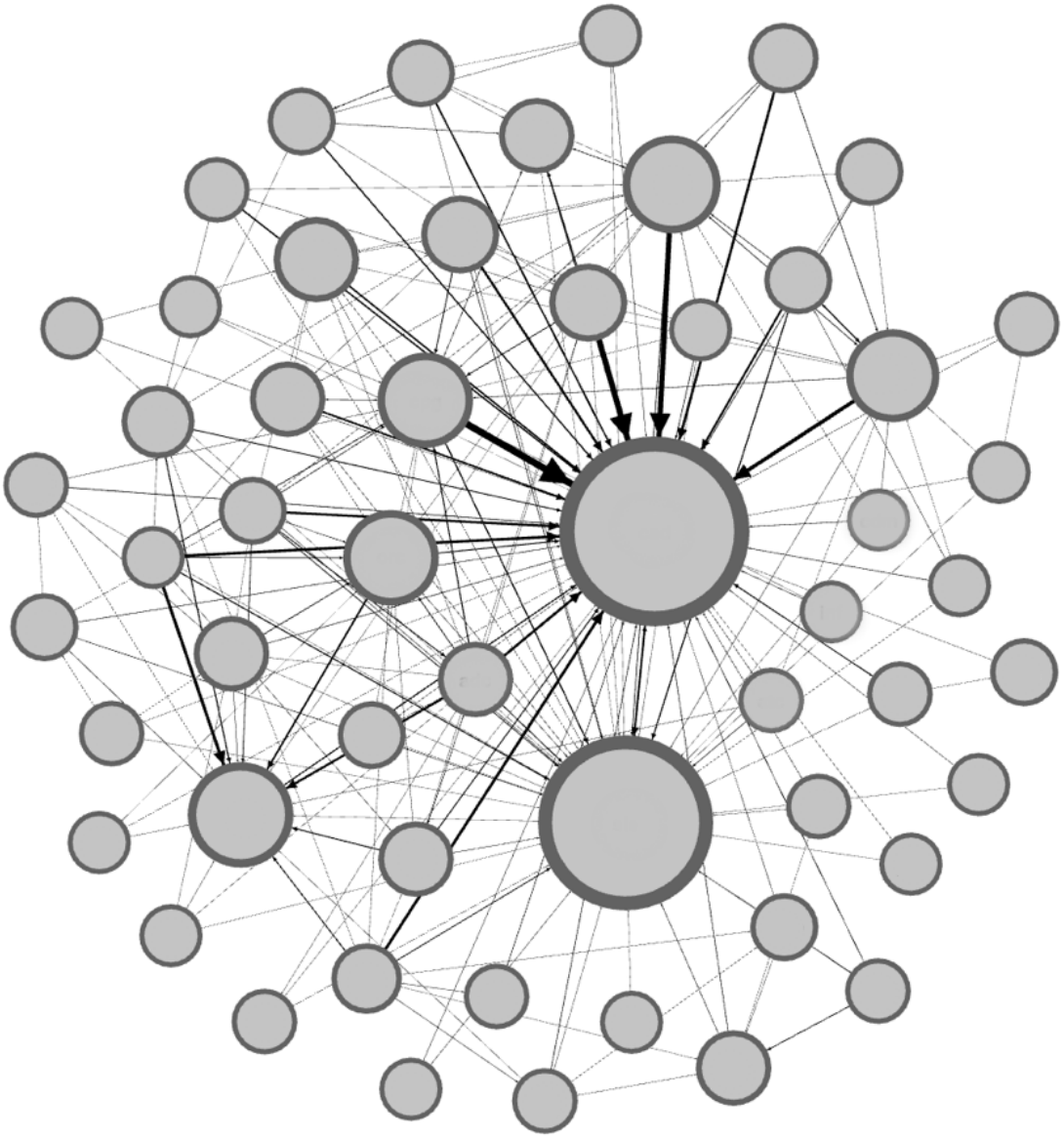


Exhibit 5
Code Complexity and Lines of Code



Source: Avalia Systems

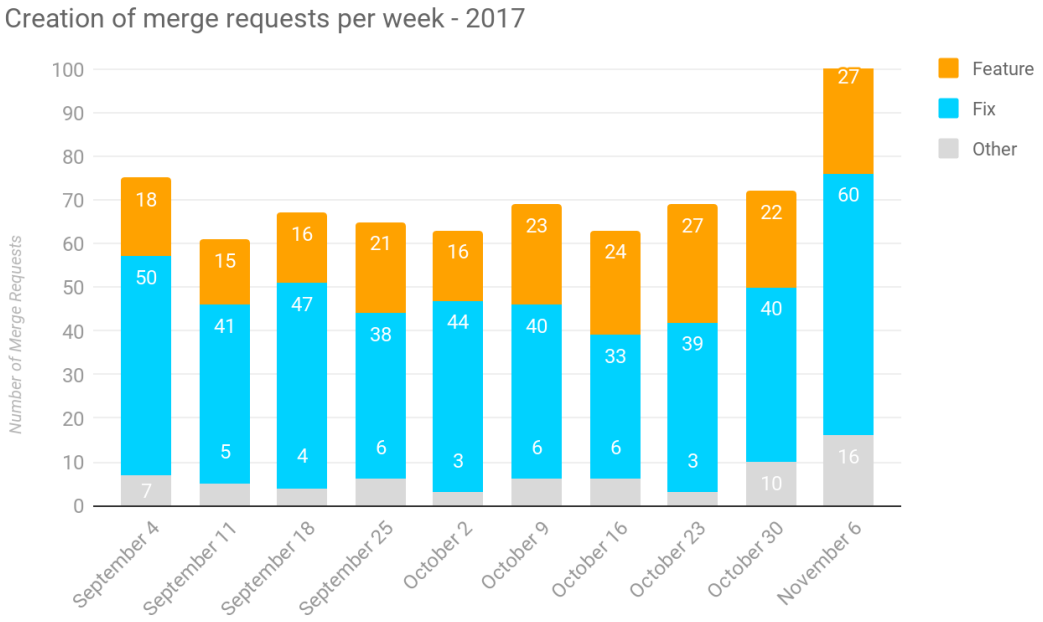
Exhibit 6
Connections between Each Module of Sienge's Legacy Code



Source: Avalia Systems

Note: Each dot represents a piece of code and each line represents a reference to another piece of code. When two parts of the software are connected, a change in one might affect the other.

Exhibit 7
Merge Requests



Source: Avalia Systems

Note: "Feature" requests refer to developing new elements of the software. "Fix" requests refer to fixing problems with the existing code.