

William E. Youngdahl

Trinisoft Technologies

It starts with one simple automation, perhaps a notification, to make things easier for users. Then more and more is added. If this goes on for too long, the underlying code becomes a big mess.

Nicolson D'Souza, Cofounder, Trinisoft Technologies

Thomas D'Souza was deep in thought. As he looked out his office window, he reflected on how far Trinisoft Technologies had come in just four years. He and Nicolson, his brother and cofounder, had just purchased a new office space in Ahmedabad, India. They added several new developers to service a growing list of clients.

Despite this success, Thomas knew that current clients provided a fragile foundation for a small tech start-up. Despite having a more diverse set of clients, Thomas couldn't quite shake the feeling that it could all disappear with one misstep. Trinisoft's fee-for-service business model meant clients would have to be replaced as development projects came to an end. Growth would require more than just replacement, and both client replacement and growth depended on tangible successes with very diverse clients.

The Early Days

Nicolson gained experience working with U.S., UK, and Europeans while working in call centers for four years while completing his bachelors and graduate degrees in computer science. He graduated with a master's degree, but jobs were scarce, given an economic recession. Many companies would require applicants to sign bonds that stipulated that employees would have to pay significant compensation back to the company if they left within one or two years. This could amount to being locked into a suboptimal job for two years. Nicolson wanted to avoid this and instead took an internship with a company that did not require a bond. He was asked to learn JavaScript, a programming language that was little more than a footnote in his computer science education.

I committed to a six-month internship position. The company needed skills in JavaScript. My classmates and I didn't think JavaScript was much of a language, but it is what the company wanted me to learn. The internship provided an opportunity to demonstrate what I could do to prove my capabilities. I quickly realized that it was quite a big thing. After just a few months, I could code at or above the levels of much more experienced employees. Just as I was feeling confident with JavaScript, some friends started talking about Elance and oDesk, two platforms that connected freelance developers with clients. They mentioned that you just sign up, enter a profile, and add hourly rates.

Nicolson was intrigued by Elance and decided, almost on a whim, to register. That very day, an entrepreneur in the U.S. reached out to him. He had an idea for an app and hired Nicolson immediately. Nicolson did not feel comfortable with the idea of leaving his day job. He was excited to have his own client, but what if that client left. Then what? He confided that he would only be able to work a few hours each day. The client accepted this constraint. Nicolson would complete the work of his full-time job at about 7:00 or 8:00 p.m. and then go to his brother's office to work another few hours.

Thomas, Nicolson's brother, also had experience working in a call center in college. One day a friend, Michael, and he were chatting and Thomas questioned why they shouldn't start their own call center. Shortly after that conversation, Thomas and Michael started their call center, just the two of them, to service two clients, one in solar energy and the other in mortgage refinancing. They rented a small office space and added a third partner. They decided to call the company Trinity, partly because of the three partners but more significantly because Thomas was a deeply devout Christian.

Copyright © 2016 Thunderbird School of Global Management, a unit of the Arizona State University Knowledge Enterprise. This case was written by Professor William E. Youngdahl for the sole purpose of providing material for class discussion. It is not intended to illustrate either effective or ineffective handling of a managerial situation. Any reproduction, in any form, of the material in this case is prohibited unless permission is obtained from the copyright holder.

While Thomas committed to being an entrepreneur after a brief discussion with a friend, Nicolson struggled more to overcome risk aversion. He described the leap of faith needed to leave his day job and a legacy of being a salaried employee:

I was never a businessperson. Neither were my forefathers. We were service employees. We worked for someone and we get a salary. But I decided it was too much to do the full-time job and the freelance work, so I made a leap of faith. Of course, for a week of work I received more than my salary for a whole year. So, I thought, even if I get paid for a couple of more weeks, it would be the same as working for a couple of years.

Nicolson provided this first client with a minimum viable product (MVP) within just a couple of weeks. The client was surprised and excited to see something working so soon. Nicolson was encouraged when the client, an entrepreneur, described trying the app for the first time as seeing his dream come true. Instead of smoothing the rough edges of the MVP to quickly commercialize his dream, the client started requesting additional features. He became more and more excited by what the app could become. Before long, the client turned the simple idea for an app into a social networking platform. He kept adding more and more features. It went on for nearly two years.

Nicolson and Thomas hired two additional developers who had recently gone through training at different companies to help accelerate development for this client. The client was excited about a seemingly endless stream of ideas that he had generated on his own without testing in the marketplace. He paid regularly, but Nicolson could see that at some point work with this client would come to an end. As much as he wanted the client to succeed, and even encouraged him to stop development and get the app to market, it became clear that the app was collapsing under the weight of its own complexity.

With the addition of two new developers, the client was paying three different individuals separately as freelancers. At that point, he suggested that Nicolson should start a business so clients could pay just Nicolson who would manage and pay the developers and not have to manage and pay separate freelancers.

As Thomas observed Nicolson working with this client and a couple of new ones, he saw more potential in application development than in running a call center:

As Nicholson started taking on new clients, I saw that there were some good opportunities. Rather than having more partners, why don't my brother and I start a company. So, we shut down the call-center business. We started working from our home. We had two or three laptops. As demand grew, Hasten, a friend and developer, joined us at our house. I realized that having three developers working out of the house doesn't look good to clients. We needed some corporate infrastructure, so I went to the market and found a small office to rent.

Once they were in the office, the brothers began hiring additional developers. After a client tried to steal one of Trinity's developers, Thomas realized that Trinity needed to start putting more structure and rules in place. The company required signed NDAs with the clients and with its developers. Thomas felt strongly that expectations needed to be clear for all stakeholders, including Thomas, Nicolson, the client, and the developers.

As revenue grew and the business was clearly viable, Thomas attempted to register the company name, Trinity, only to discover that the name was already registered by another Indian company. Thomas and Nicolson settled on the name Trinisoft Technologies. The venture became official in 2014, but they were still on the steep portion of the service delivery learning curve.

Working with Clients

Nicolson worked directly with clients and the team of developers servicing them, while Thomas focused on the business and financial management of Trinisoft. Nicolson described the approach and challenges that he and his team faced in working with clients:

The first thing we ask when looking at client requirements is whether meeting the client's requirements is technically feasible. Can we build such an app? The next question we ask is whether the app can

be developed in an efficient manner. I'm talking about the code itself. Is it efficient? To make an app easy for the user, we often add more technological complexity. It starts with one simple automation, perhaps a notification, to make things easier for users. Then more and more is added. If this goes on for too long, the underlying code can become a big mess and the app becomes buggy and slows to a crawl.

In addition to fighting complexity, Nicolson faced other challenges with clients. One client approached Trinisoft with the idea of building a better version of Google. Despite an unrealistically low budget, Nicolson and the developers were excited by the technological prospects. After a month, the developers built something that was about 10% of what this client wanted. At that point, the client said that he would not pay anything until he received the complete app, 100% done. Thomas and Nicolson had to tell the client that it would not be possible to continue under those terms. After a month of development, Trinisoft did not get paid.

The next client paid promptly for completed work, but every five minutes he would ask, "Is it done?" He insisted that Trinisoft developers use screen-shot tracking software so he could monitor work. If he saw that a developer had not moved a cursor in 10 minutes, he would contact Nicolson to question the efficiency and progress. The client did not consider that a developer may not have moved the cursor because he was reading the screen and thinking about the best solution to a coding challenge.

At that point, Trinisoft did not have other clients, so letting go of a difficult client who paid well was not an option. Nicolson and Thomas also believed that they would learn from clients that weren't so easy. It would make them and their company stronger. They would have to learn to work with clients who have different tastes, different styles, and different requirements.

Once they landed a new client who was also paying well, it became increasingly clear that the difficult client was consuming all of Nicolson's time despite paying for only the developer's time. He simply could not provide adequate service to the new client. Nicolson's frustration became very hard to mask:

So here was a client who had paying capacity. And he did pay. But still, we realized that we could not keep him. We couldn't even respond to requests from potential new clients. And this new client was paying us better and did not taunt us every minute.

Letting go of this difficult client was a bittersweet event for the development team. Thomas and Nicolson had refused business from prospective clients, but this was the first time the company had essentially fired an existing client. The move freed Nicolson to support the new client and to help Thomas pursue additional clients. It also lifted the spirits of the developers who were feeling henpecked by the micromanaging client.

Discovering a Better Way to Build an App

Trinisoft had successfully developed several apps for entrepreneurs, but none of these individuals followed through to take their apps to market. One client told Nicolson, "I have the app in my back pocket." Potential clients wanted to see apps that Trinisoft had successfully built for other entrepreneurs. It was very frustrating for Nicolson and Thomas to not have an app that they could point to in Apple's App Store or Google's Play Store.

In 2015, a client approached Trinisoft with a request to help fix an app that started as a simple sticky-note brainstorming app but cycled through two years of feature-bloating complexity addition. Nicolson discouraged the entrepreneur from continued development of the app, but the client was emotionally attached to the project and asked Nicolson to spend a couple of weeks to see what he could do. During that period, the server went down on a Sunday morning. With a small but loyal base of active users, this was the worst possible situation for the client who described the event:

I was in Prescott, Arizona. Nicolson was in Ahmedabad, India, 8,441 miles away. It was 12:30 p.m., Sunday, in Prescott and 1:00 a.m., Monday, in Ahmedabad. Instead of waiting until the morning when he reached the office, Nicolson used his mobile phone (with a 2G connection) to take control of my Mac. He then loaded Meteor and Node.js onto my computer and walked me (the non-technical founder) through the steps to deploy a code fix.

After two weeks passed, Nicolson told this new client that it would take a complete rewrite of the code to debug and simplify the app. Instead of encouraging this path, Nicolson presented an interesting proposal:

I encouraged this client to let me build something better. I asked him to give me just 30 days to build a better app. This client had learned from his mistakes, and we had learned from the mistake of letting clients pursue what we call “loops” that add nothing but complexity. He had a new idea for an app, and we committed to keeping the app as simple as possible.

Nicolson and the client committed to minimizing complexity. The client insisted that Nicolson challenge him on any requirement that could increase complexity. Nicolson committed to teaching the client as much as possible about what drives complexity in the underlying code. The non-technical entrepreneur was eager to learn, given his passion for app development, and he knew that spending six figures creating complexity was not a viable path. If the MVP could be developed for \$2,000 in a month, it would be nothing short of a miracle to the client.

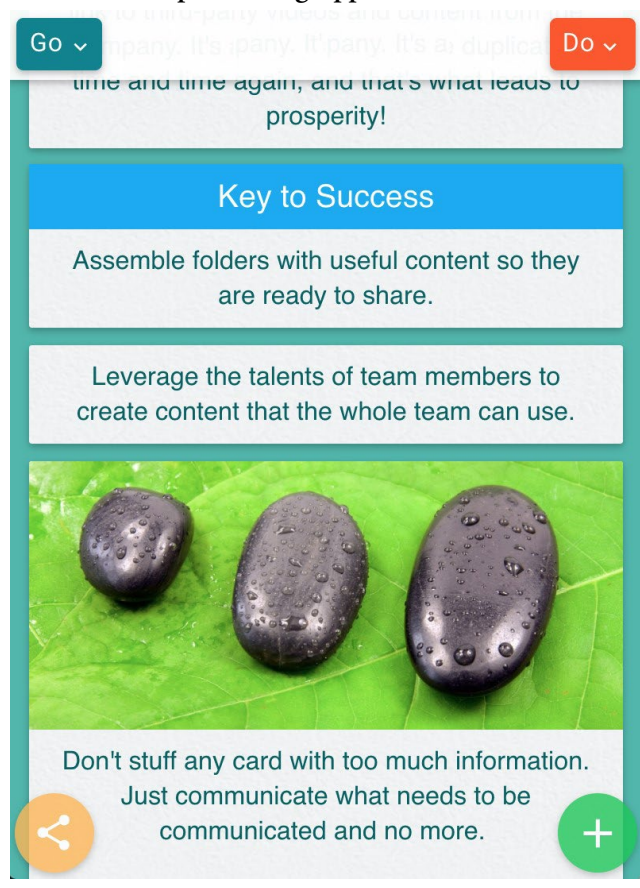
The idea for the app was simple. It would give network marketers the ability to organize and share company information, and their own content, as well, with prospects, mobile-to-mobile. Instead of sharing several links in an email, a user could organize information into cards that would be ready to share. The client’s wife was a network marketing professional who needed this capability for her team, so the problem was well-defined and a team was ready to test the app and provide input, as needed, to the development process.

Without prompting, the client supplied a detailed mockup to show the desired user interface (see Exhibit 1 for an example). Different PowerPoint slides were created for each possible function and view of the app. This non-technical founder did not want to repeat prior mistakes:

I suffer from adult ADD and love to ask “What if...?” daily. But in adding teams, spaces, commenting systems, notifications, lists, a WYSIWYG editor, file management, hand drawings, and a Trello-like mashup, we super-glued complexity into what should have remained simple. The code became a black forest of complexity. The time to define the end is at the beginning of the project. How we get there can flex, but the destination is a fixed dot on the horizon.

Written documentation and the mockups made the endpoint very clear, but coding the user interface would have presented a challenge if it were not for the use of a template. Based on experience working with a prior client who used templates to significantly accelerate development, Nicolson encouraged his client to select a template.¹ A template provides the basic look and feel of the app, and contains specific components that can be used by developers to create specific capabilities. A simple example would be a drop-down menu that simply requires new descriptions and links. In addition to providing the basic elements needed to accelerate development, these templates were mobile-friendly, or responsive. They were designed to look right on mobile devices while also being acceptable on computer screens.

Exhibit 1. Simple Sharing App for Network Marketers



¹ See this link for examples of templates: <https://colorlib.com/wp/free-html5-admin-dashboard-templates/>.

Nicolson encouraged the client to select a template that used semantic UI because Trinisoft developers had experience using this type of template. After a couple of days, the client communicated a strong preference to go with a template that used materializecss based on Google Material Design.² Material Design had the now-familiar cards (see Exhibit 2) that can be seen on many websites. Nicolson debated challenging the client, but Google Material Design was getting a lot of traction in the marketplace. Additionally, it was skill that his team could be paid to learn and then apply to future clients.

Nicolson and the client spoke on a nearly daily basis to assess progress and discuss any trade-offs that would increase complexity. For example, a decision was taken to add a to-do list, but after meeting with beta testers, it became clear that this was not a valued feature and was removed. After just 30 days, a minimum viable product was ready. Two weeks later, the app was in both the App Store and the Play Store. Days later, the first sales started to register at \$7.99 each. Nicolson communicated his pleasure in having an app to highlight when communicating with potential clients:

This completely changed our ability to promote Trinisoft's capabilities. The app was built so well that we could showcase it during initial discussions with potential clients and we would usually get their business, at least when they wanted to build a mobile app.

Managing Growth

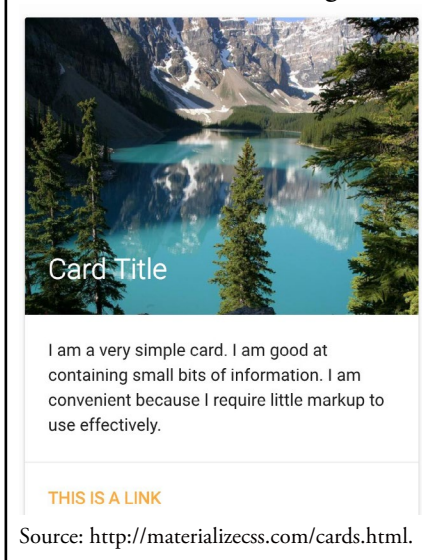
Trinisoft grew to 10 employees in 2015. This approached the upper limit of the rented office space. Thomas spoke with Nicolson about the possibility of purchasing their own office space. In 2016, they acquired an office in Ahmedabad that could accommodate 30 people. By the end of 2016, their headcount was 15.

As Nicolson stayed very busy servicing clients, Thomas was shifting more of his attention to Trinisoft's business model. Having to replace clients as a natural consequence of development cycles consumed a great deal of time and attention. Growth required replacements, plus net additions to the client base. Would word-of-mouth suffice, or would the company need to begin focusing more on sales and advertising? To increase capacity, Thomas was considering 24-hour work schedules. The developers could do their work on India time while those communicating with clients could be working on U.S. and European schedules. But would this be enough, or would new business models be needed? Thomas had some initial thoughts:

Looking to the future, perhaps we will find ways to partner with clients to ensure a service contract for several years. We could also develop our own products and then offer both product and service. It's also possible that someone will come to us with a very good idea and we could become a developing partner. We would have to find the right profit-sharing or equity arrangements for this kind of partnership.

Just as Thomas was thinking about how Trinisoft might engage in different types of partnerships, Nicolson pinged him while on a Skype call with a potential client. "Thomas, more business. ☺" Thomas quickly typed, "Excellent news. Let's meet for an hour before heading home today." After tapping the send key, Thomas walked to a white board and began capturing thoughts to share later that day with Nicolson.

Exhibit 2. A Material Design Card



² See this link for the selected template: <https://themeforest.net/item/materialize-material-design-admin-template/11446068>.