Chapter 4

Fueling Learning with Sugar

“We're too concentrated on having our children learn the answers. I would teach them how to ask questions—because that's how you learn.”—David McCullough

From the time of OLPC’s founding we were focused not only on designing and building a laptop. We also recognized that what children do with their laptops matters. For all of the publicized focus and internal excitement about the potentially large numbers of laptops to be deployed, the engineering team was equally as focused on how the laptops would be used for learning. The questions we asked ourselves repeatedly were: Is the laptop capable of driving the learning of a given child? And does our program model drive the type of change that we want to realize in the culture of education in developing nations?

In his 2006 TED talk,[i] Negroponte made a public promise: he indicated that OLPC would not launch until we had commitments for sales in excess of 5 million units. Second only to the $100 price tag, the number of XO laptops that were actually delivered into the hands of children worldwide became a publicly referenced metric for gauging the organization’s success over time. While the goal of one-to-one computing—i.e., a laptop in the hands of every child—had always been part of OLPC’s vision, the pressure to “hit the numbers” pushed the organization to focus almost exclusively on the manufacture and distribution of laptops.

While Negroponte reinforced that OLPC is “an education project, not a laptop project,” in nearly the same breath he emphasized that OLPC would not provide the educational software to enable learning, but would instead look to the countries in which the laptops were deployed to do this. The relative emphasis on achieving the goals of one-to-one computing (driven by the metric of scale) versus driving children’s development (driven by learning metrics) was a dichotomy that would come to define our future (and became the cause of a rift in our organization that would eventually take us in different directions).

The story of creating learning software for the laptop – which we later named Sugar -- is largely untold. The development of Sugar began in 2005 when Walter Bender—then executive director of the Media Lab and a member of the original OLPC team—began a software initiative specifically aimed at increasing the probability of children learning with their XO laptops. The development of Sugar closely parallels the development of the XO hardware: the software and the hardware were developed concurrently, initially by the same team of designers and engineers, working under the enormous pressure of limited time and resources.

An Electronic Play Space

When we hear the term “educational software,” most of us think of a colorful interface of games and exercises designed to deliver a specific curriculum or body of knowledge that has been defined as important for children to master. Even interactive educational games, such as Number Crunchers, a number-puzzle challenge game created in the 1990s and often used to “teach” math facts in schools, are usually little more than animated worksheets trying to disguise the fact that children are merely answering questions by rote. The core idea driving Sugar is the notion that children and their teachers can do more than just use a computer for answering questions; they can use the computer as a “thing to think with” to spark *real* learning. Sugar turns the traditional educational software paradigm on its head, replacing interaction between children and rigidly defined electronic educational materials with a set of tools that encourage creativity and enable even very young children to program and modify those tools.

One of the key opportunities in developing Sugar was the chance to invent a new paradigm for human–computer interaction. Alan Kay once defined technology as “anything invented after you were born.” This definition reminds us that children do not have any set expectations for how to interact with a computer—they readily adapt to what they are using and they have no innate affinity for the “Windows paradigm” to which most of us are accustomed. With Sugar, there was an opportunity to start fresh and revisit many assumptions, to break free of an existing paradigm and start over. Out with the old: overlapping windows, left and right-click distinctions, and double-clicking; and in with the new: applications running full-screen, hover menus, interoperability and data exchange between activities. The software designers were free to be creative on a whole new level.

On the surface, Sugar is just another desktop, albeit with a different look and feel. But there are things that Sugar enables that no other computer you have ever worked with does. Sugar presents the user with a different type of “space” that feels more like a journal or a sketchbook than a filing cabinet. Sugar is designed to take away everything that sits between you and the information you are interacting with: there are no files, desktops, or passwords. In some sense, Sugar is much like a kindergarten classroom, in which a child can move fluidly between the block area, the trucks and cars area, the art area and the playhouse area. Likewise, in Sugar, there is spillover from one activity to another.

Learning through Doing

Sugar focuses on helping children to make knowledge their own by putting it to work on problems that are meaningful to them. As a consequence, learning is something done *by* the student rather than *to* the student. In this type of learning, a child will not simply know the name of a thing and how it might be used in a narrowly defined context; they will understand its utility and its limitations. Put more succinctly, “You learn things through doing, so if you want more learning, you want more doing.” This is not a new idea: much of the groundwork was laid by John Dewey more than a century ago in his work on experiential education and hands-on learning that emphasized active participation by the learner.[ii]

In keeping with the notion of creating an electronic play space, Sugar is a collection of “activities,” rather than a set of ready-made applications. These activities compel the learner to take action: to create something and to share the creation with a peer, a teacher, or a parent. To reinforce the idea that to learn is to do, each activity is named with a verb. Standard applications include those that will feel familiar to anyone who has used a computer: Write (a word processor), Browse (a web browser), Paint, Calculate, Record (a multi-media recorder), Read (an e-book reader), Chat (real-time messaging) and Speak (a voice synthesizer that allows you to type something in and then repeats it back to you).

The XO as a Sonar Device

Not long after the first version of Sugar was released, Ben Schwartz, a Harvard student, stopped by our office to borrow two XO laptops. He wanted to test a Sugar activity he had written that measures the distance between two laptops. By combining the capabilities of the microphone and Wi-Fi radio built into the machine, he was able to use the laptop as a sonar device. His breakthrough application was aptly called “Distance.” This creativity led to unexpected activities: for example, in Brazil the program was used in gym class to calculate the median height of the students and map their school.

Reflective Learning

Education experts agree that the best approach to learning involves doing and then stepping back to reflect on the doing: What did I learn? How can I use that? What questions do I have?[iii] By helping children to ask good questions about the things they have done, as opposed to remembering the right answers, we are helping them to build the critical thinking skills that enable them to be independent problem solvers. Without reflection, learning is an open loop, and an open-loop system can neither identify and correct errors nor adapt to change.

Sugar facilitates reflective learning by ensuring that everything children do with their computers is recorded in individual journals that include screen captures of their work. After every activity, children are encouraged to write their observations and reflections, which are also saved in their journals. From these records of activities, children can create portfolios—multimedia narratives that show what they have done, how they did it, and what their thoughts are on what they have created. Children essentially become curators of their own work. Engaging children in telling about what they have learned as a “story” is a simple way to help reflection become a norm in their education.

By building upon the automatic accumulation of work in the Sugar journal, the portfolio process can readily be integrated into the classroom routine. It can be used as an assessment tool to help teachers, parents, and school administrators understand better the depth and breadth of what a child has learned.[iv] At a “portfolio social,” parents can be invited to view presentations and ask children about their learning. The classroom teacher can add additional assessment slides to the portfolio addressing themes such as work habits and personal growth. This can become part of an archive that travels with a child across grade levels. Through the juxtaposition, the child and teacher can see what has changed over the course of the years, what trends are, and what areas need improvement.

In addition to the journal feature, Sugar applications are enhanced by specific features that help children to collaborate, sharing information, insights, and discoveries, solving problems together and co-creating. Write, for example, has a feature that allows peers to edit an essay or story a child might compose; Browse allows children to share bookmarks for pages they find interesting with other students; Record allows them to share photos in real time; Turtle Art lets them program Logo turtles within the same workspace. Interestingly, in many traditional school environments, this collaboration capability might be called cheating. For Sugar, in contrast, working together is a fundamental part of how children learn and is something to be embraced.

Understanding in More than One Way

Sugar activities encourage children to explore multiple ways of understanding a new concept or distinction. This is fundamental to real learning as opposed to memorization. As Marvin Minsky remarked, “Until you understand something more than one way, you don't really understand it.”[v]

In fact, the very recognition that there are multiple ways to solve problems gives learners a deeper understanding of the world around them and a stronger sense that it is possible for them to develop a new way to approach something. Sugar gives the child the ability to seamlessly connect ideas and efforts across domains. It is idiosyncratic and personal and specifically designed to promote the cross-pollination of ideas that is so critical to creativity.

Opening up the Black Box

Forty years of working with children and computing has demonstrated that engaging children not only in using computers, but in programming them is a powerful means of driving learning. The process of writing and then repairing or "debugging" a program—which Cynthia Solomon (one of the inventors of Logo) described as "the great educational opportunity of the 21st Century"—provides a basis for active learning through trial and error.[vi] In Sugar there are no black boxes: the learner sees what something does and how it does it. With just one keystroke, the Sugar "view source" feature allows the user to look at any program they are running and modify it. The premise is that taking something apart and reassembling it in different ways is a key to understanding it.

Invent Your Own Abacus

As most of us know, an abacus is a tool constructed of a frame with sliding beads that is used for performing arithmetic. Sugar has created a number of different types of abaci to help children grasp complex mathematical concepts easily. It also lets children design their own abacus. Teachers in Caacupé, Paraguay, were searching for a way to help their students with the concept of fractions. After playing with the Sugar abacus activity, they conceived and created a new abacus that lets children add and subtract fractions. Sugar didn't just enable them to invent, it encouraged them to invent.

Sugar also comes with a variety of programming environments that allow children to use a computer as a tool for creativity. These include many now well-known programs such as Turtle Art, which is based on Logo and allows children to make images while learning concepts of geometry and programming; Scratch, a programming language that enables the creation of interactive stories, games, music, and art; and Squeak Etoys, a multimedia authoring environment and visual programming system that is meant as an educational tool to teach children powerful ideas (primarily science and math) in new ways.

The ability to not only *learn with the machine and software but also to manipulate and change the software and hardware* itself opens the door to learning lessons far more important than those necessary to pass a test. It leads children to the discovery that they are what David Cavallo would term “authentic problem-solvers” in the real world[vii]. Success at fixing a program also gives students confidence that they can apply the same skills—defining problems, developing hypotheses, creating tests, and executing solutions—to other problems they may encounter.

Developing Sugar

Like the XO, much of the early development of Sugar took place in the MIT Media Lab. Development of Sugar began in the spring of 2006, in parallel with the work of the teams responsible for developing other aspects of the XO laptop's software, including device drivers, power management, and security. One might ask how OLPC—an organization of fewer than twenty people that had already taken on the daunting task of building the XO laptop—was able to create an entirely new learning platform out of whole cloth, and do so with almost no investment in software engineering. The short answer is that they didn't. OLPC solved the problem of how to develop the Sugar software with limited resources by attracting external resources—not creating them from scratch—while articulating a very clearly defined objective and set of measurable outcomes. OLPC built upon decades of research into how to engineer software to promote learning and amplified OLPC's staff resources by leveraging key partnerships within the free software movement.

Our principal partners in Sugar development were a small engineering team from Red Hat, a major distributor of the open-source computer operating system GNU/Linux, and Pentagram, an international design partnership that does work in graphic design, identity, and product development. The Red Hat team, under the leadership of Chris Blizzard, an experienced systems engineer, was tasked with leading the software engineering effort behind the development of the Sugar desktop. Lisa Strausfeld, a former MIT Media Lab student, led a team from Pentagram tasked with developing the interaction design and graphical identity of Sugar. In six months, this core group was able to produce a basic framework for Sugar upon which a community of pedagogists and software engineers could build learning activities.

Both Blizzard and Strausfeld joined the team in large part because of their personal enthusiasm for the OLPC mission. Each wanted to bring their skill sets to the project, but also, each had personal ideas about pedagogy that they wanted to express in the project. Blizzard, who has a background in free and open-source software, wanted to bring some of the culture from that community into the classroom. Strausfeld, who studied with Muriel Cooper at MIT, was interest in exploiting the potential of children learning through engaging in design practice. The opportunity to help shape the project from a pedagogical perspective was a strong motivator for every one of the early participants and remains an important motivator for participation even six years into development.

Rather than try to build everything themselves, OLPC worked with the free software community to leverage many preexisting software projects. This community is made up of a large group of computer programmers who are committed to an ethic of user freedom to run, copy, distribute, study, change, and improve software. The community openly shares programs they have written with one another. Any member of the community has the right to use a program as-is, or to adapt it as desired, without needing permission of any sort. The key challenge for OLPC was working effectively within this community software–development context without losing sight of the learning mission.

Sugar drew inspiration for its activities and the fluid interface between activities from observing how the free software community collaborates. In this community, software developers chat, socialize, play games, share media, and collaborate on media creation and programming in both formal and informal settings. The Sugar parallels to the free software movement are tools of expression, children creating content as well as consuming it, and a strong emphasis on collaboration, co-creation, and helping one another. As with free software, Sugar encourages every child to be a creative force within their community. Remarkably, approximately 10 percent of all Sugar activities have been written by preteen-age children!

Now We Have Hackers!

When the current president of Uruguay, José "Pepe" Mujica, learned that a twelve-year-old from a small town east of Montevideo had programmed six entirely new Sugar activities for the XO, he smiled and said triumphantly: "Now we have hackers." In his eyes, this one child's ability to contribute to the global Sugar development community was a leading indicator of change and development in the country.

With the engagement of the core development team and a growing number of volunteers from the free software community, it took only six months to move from idea to completion of the first version of Sugar. The team used an iterative-design process: rapid prototyping of ideas, followed by critiques, followed by coding. We went through two to three cycles per week until we reached consensus on a basic framework. It was at this point that we were able to set higher-level goals enabling participation by a broader community of developers.

Like the XO development process, which was going on in parallel, the software development process required ongoing efforts to solve knotty and often unprecedented technical problems. To wrestle with these, the OLPC, Red Hat, and Pentagram teams met face-to-face on a biweekly basis. The broader development community, which was dispersed across five continents, was engaged in addressing the same problems and met 24/7 in multilingual online chat forums. This was a global movement: the lead developer lived outside of Milan, Italy, the lead community contributor lived in Siberia, and the principal testing team operated out of a coffee shop in Wellington, New Zealand. Significant contributions were made by a high-school student from Wunstorf, Germany, an energy-management consultant living in Melbourne, Australia, and a student at the University of San Carlos in Brazil. The use of modern software-development tools, such as distributed source-code management and wikis enabled members of the development community to collaborate anywhere and at any time. We were also able to pilot Sugar in a wide range of contexts, getting hands-on experience and feedback in schools in Nigeria, Thailand, Cambodia, and Brazil.

Sugar was designed so that new uses emerging from the community could easily be incorporated. The journal was the brainchild of Ivan Krstić, who also designed OLPC's security model. Popular activities came from community volunteers such as Brian Silverman, a long-time collaborator of Papert's who created Turtle Art, and Alan Kay and the Viewpoint team who created the Etoys learning environment. Others were commissioned from specific individuals, including a multimedia activity called Record written by Erik Blankinship and Bakhtiar Mikhak; the Sugar word processor, Write, which was based on Abiword and written by J. M. Maurer; the TamTam musical activity suite written by Jean Piché and his students at the University of Montreal; and some constructionist games from Harel's MamaMedia group that were "sugarized" by Morgan Collett and Carlos Neves.

Sugar was explicitly designed by OLPC to be augmented and amplified by its community and the end users: once these initial examples were published, the floodgates opened and activities began to come in unsolicited. While we had the advantage of a highly publicized project—OLPC was the subject of almost daily international news coverage—we did not necessarily have direct access to the highly skilled software-development community we needed in order to grow. We therefore did outreach in the forums where these people hung out. In free software, that is primarily in chat rooms and at conferences. Blizzard and the Red Hat team established an IRC channel for the project that soon attracted nearly one hundred concurrent users. Gettys spend a great deal of his time attending free software conferences, focusing especially on conferences in regions where OLPC was targeting deployments, in order to solicit volunteers. We also used word of mouth, leveraging both the MIT alumni network and friends and colleagues from industry.

Sugar was also designed so that end users could make contributions to the software itself. One of the stories that exemplifies this took place in Abuja, a city in Nigeria that became that nation’s capital in 1991 (as a neutral location it was hoped that it might placate some of the ethnic and religious divisions that have beset the county). Abuja was the site of the first OLPC pilot for the beta XO laptop and Sugar. While teachers and students took to the laptop quickly in general, they did confront some problems. The most notable of these was that the Write word-processor activity did not have a spelling dictionary in Igbo, the dialect used in this particular classroom (and one of the more than three hundred languages currently spoken in Nigeria). From a conventional software-development standpoint, solving this problem would be prohibitively expensive, demanding huge resources. But for children in Abuja equipped with Sugar, the answer was relatively simple. Confronted with the problem of lacking a dictionary in Igbo, they made their own Igbo dictionary. The free software ethic built into Sugar enabled local control and innovation.

By the end of 2006—less than six months after the idea was born—Sugar had a basic system running that included all of the basic activities: Write, Browse, Read, Paint, etc. By the end of 2009, Sugar had hundreds of activities contributed by thousands of developers around the world, and the ongoing engagement of a global group of developers, teachers, and students.

A Fork in the Road

Despite the fast progress and impressive accomplishments of the team that created Sugar, within OLPC there were real and growing concerns about creating a new operating system for the laptop and resistance to expending resources on its development. At the same time, XO laptop sales were failing to meet expectations. While the development process for the XO resulted in the first machine being finished and ready to ship in November 2007, the original partnerships that we had counted on for the target 5 million laptop sales had not materialized. Instead, by the beginning of 2008, OLPC had commitments from Uruguay, Alabama, Peru, and Mexico that totaled only 425,000 machines. Being so far below projections was putting some pressure on the organization. We had been funding development with membership fees from the OLPC industry consortium; even with those funds taken into account, we had been sustaining a burn rate that exceeded our income. Our forward spending was with the expectation that these costs would eventually be balanced out by funds received from deployment contracts.

While we did not panic when confronted with a slow sales cycle, we did begin to look for explanations. Some of the fallout was the result of events beyond our control: a coup in Thailand, a personal tragedy in Pakistan. Some was due to lackluster commitment on behalf of our partners: President Olusegun Obasanjo of Nigeria, presumably looking for excuses for inaction, said he would buy the computers for Nigeria only when they cost $100. Further, competitors from the commercial sector were working hard to undermine our efforts, most effectively in Brazil and Argentina. Nonetheless, we were concerned that while our mission was well received for the most part, our product was not meeting the needs or expectations of the market.

The OLPC board was not convinced that investing in a custom and unique software user interface was a realistic way of supporting OLPC's goal of shipping large numbers of laptops. They had serious doubts about the ability of OLPC to penetrate the education market without offering a more conventional approach to software. For governments that considered adopting the notion of one-to-one computing a big step, adding the uncertainty of unfamiliar software made the decision all the more difficult. Fueling the tension was a report from a meeting that OLPC had with the Egyptian minister of education, in which he was asked if Microsoft Windows could be run on the XO laptop (the answer at that point was no). Facing such requests, Negroponte announced at the February 2008 OLPC board meeting that the organization would refocus its efforts around Windows. Much of the board breathed a sigh of relief, as they predicted that switching to Windows would mean significant growth in the number of laptops deployed.

Meanwhile, market demand for cheaper, lighter laptops—netbooks—was growing and that industry was gaining momentum. While OLPC was the market leader, it was clear that there would be many competitors emerging in the near future, all of which had potential as platforms for educating children. The Sugar community and many at OLPC considered the question: "If we believe that Sugar offers a great learning experience, why not make it available to children regardless of which computer they are using?" This was a clear instance of divergent theories of change.

After the February 2008 board meeting, OLPC and the Sugar community split. One month later, Sugar Labs was founded as an independent non-profit organization that would continue to drive the development and dissemination of the Sugar learning software. In the immediate aftermath of the split, OLPC rapidly expanded its staff, going from fewer than twenty full-time employees to nearly sixty. However, only a few of these were actively working on user-interfacing software. Sugar development did not lose any of its momentum, as the volunteer community continued its efforts and the handful of OLPC engineers who had been working on Sugar—some of whom were founding members of Sugar Labs—pressed on as well. Meanwhile, Negroponte was convinced that a switch to Windows would result in hundreds of millions of laptop sales. Bender was convinced that Sugar would have value even without the OLPC-installed base. At the management level, there was a tacit agreement to go our separate ways.

Tension at OLPC did not diminish after the split with Sugar Labs; there was a continued internal debate between lofty aspirations and tactics. One year later, there was a further split that resulted in the creation of the OLPC Foundation and the OLPC Association as separate entities. The Foundation, run by Negroponte, was tasked with the “big picture” and future thinking. The Association, run by Rodrigo Arboleda, was tasked with the practical work of driving impact, defined as getting the laptop built, getting tools onto the laptop to enable learning, and getting the laptop into the hands of children and their teachers.

It is not unusual that the founders of an idea or movement have different ideas and that their paths diverge. This is just as likely to occur in a social enterprise as in a commercial enterprise. While breaking up is hard to do, its consequences can be positive. The anticipated boost in sales from Windows never materialized, and OLPC still represents 95 percent of the Sugar user base. Three years after the split, despite having had differences of opinion about tactics, OLPC Foundation, OLPC Association, and Sugar Labs all work together closely, each playing a unique role in advancing the opportunities for learning from independent but synergistic perspectives. OLPC Foundation drives future hardware development; OLPC Association builds and distributes laptops; Sugar Labs provides the software that is run on those laptops; and the OLPC and Sugar communities work together with local deployments to support children and their teachers.

Sugar Labs, Plural

In the aftermath of the split with OLPC, creating a new organization to develop and maintain the Sugar software was straightforward. The founding members consisted of a small core of developers who were dedicated to the project, including some members of the original OLPC engineering team. The Sugar community suffered very little attrition after the split with OLPC; rather, when Sugar Labs was created, some new developers joined who were inspired by the new direction. Sugar Labs is a member project of the Software Freedom Conservancy (SFC), a not-for-profit organization that helps "promote, improve, develop, and defend" free software projects. The SFC provides a non-profit home for Sugar Labs and it takes care of any project needs not directly related to software development. The decision to move the project to the SFC was in large part motivated by the potential for networking provided by alliance with other free software projects. Only by building partnerships has Sugar Labs been able to operate with a balance sheet of zero revenue and zero cost.

The real challenge that we faced was how to create an organization that would support our learning mission, enable ongoing innovation, and allow for local contextualization and appropriation. The key innovation that Sugar Labs embraced was creating a flexible model that supports and encourages the creation of local and regional labs. The local labs take whatever form makes sense in each country's or community's context, including, in some cases, being established as for-profit enterprises, which Sugar Labs "central" cannot do. These labs were originally encouraged in those countries in which the XO laptop was being deployed, to ensure that Sugar meshed with the local culture and to give local communities a channel for contributing to Sugar's ongoing evolution. Sugar Labs affiliates exist in most of the countries in which OLPC has major deployments—Uruguay, Argentina, Paraguay, Peru—and in some of the countries in which smaller numbers of laptops have been deployed—such as Colombia and Chile. Local lab members are on a par with their global colleagues: everyone learns to cooperate, compete, and contribute in a global marketplace. There are also Sugar Labs that exist without direct affiliation to a country-level deployment, such as the Washington, DC, lab, which rallies its members to support Sugar from technical, pedagogical, and marketing perspectives.

These regional labs play a number of important roles in supporting Sugar's success. The first is providing for communication between local communities and the global developer community. The constant flow of information back and forth between the global and local Sugar labs is mutually beneficial, reinforcing the mission and core principles and circulating new ideas and new energy as new challenges are encountered. Regional and local labs also develop new activities for Sugar; overall, local labs are the source of an increasingly large percentage of all Sugar activities.

When working with local groups, two questions often arise: how to recruit and how to ensure quality. In the case of Sugar, the recruitment occurs through multiple channels: there is interest driven by the OLPC deployments; local free software developers are often looking for a local outlet for their talents; and local universities are looking for ways to both engage their students in social action and provide tangible outlets for their engineering students. Quality control operates in the manner of all free software projects: software is submitted "upstream"[viii] to the maintainers, who vet it for quality. High-quality work gets absorbed into the project at a global level. But whether or not software developed locally is suitable for deployment is ultimately a decision made by the deployments themselves. The structure of Sugar is such that local repositories of activities (e.g., the local “App Store”) are on an equal footing with the global repository.

Fueling Entrepreneurism with Sugar

In Uruguay, Plan Ceibal was developed to deploy one laptop per child throughout the entire country. A by-product of this initiative has been the creation of new jobs: everything from logistics and technology support to the installation of power and network infrastructure to teacher training and curriculum development. For just this one educational initiative, more than three hundred new jobs were created. And not just for local consumption: Plan Ceibal personnel consult internationally; for example, they helped design and implement a deployment in the Nagorno-Karabakh Republic.

Sugar development began in 2006 under the OLPC umbrella. It was spun out as an independent non-profit, Sugar Labs, in 2008. The lessons of Sugar's development for non-profits and social entrepreneurs are rich. We have a balance sheet of zero revenue and zero cost and a 100 percent volunteer staff numbering in the thousands. We have consistently been producing a new release of the software every six months. Our software is available in more than twenty-five languages and is used by more than 2.5 million children in more than forty countries each day. These statistics are impressive for any software development organization, let alone a volunteer-led non-profit foundation.

Lessons and Reflections

The experiences of Sugar Labs, including its parting of the way with OLPC, provide some object lessons for social entrepreneurs.

- *Necessary tension between organizational sustainability and widespread impact.* Social enterprises start out by focusing on a social problem and desired impact and create an organization to serve as the tool for driving impact. Over time, it is not uncommon to focus on the sustainability of the organization just as much as the end-state impact about which you care. In the case of OLPC, the original public promise to achieve the $100 price point and deliver millions of laptops—which had been so effective in garnering attention and resources—grew to overshadow the impact on learning that was the original focus. In the case of Sugar, the choice to make the software freely available to anyone with a computer had the potential to dilute a key strategic advantage of OLPC in the marketplace. On the other hand, it also created the opportunity to drive impact in a broader set of countries where the XOs were not deployed. By embracing Sugar Labs, the more narrowly defined interests of the OLPC Association, which aimed to maximize the number of computers sold, evolved into a solution that could again focus on learning. This tension of balancing the requirement to create a sustainable funding base with maintaining focus on addressing a system-level social need should be familiar to any social entrepreneur.

- *Balancing control and autonomy.* Deploying Sugar through the global-local Sugar Labs networks provides a promising model for how social entrepreneurs can successfully scale their efforts. By creating a structured platform and approach to development, Sugar effectively created "guardrails" that allowed for local customization and innovation that would contribute to the overall mission. The deliberate effort to hand over control (and responsibility) for Sugar to local and regional groups led to self-determination and local relevancy. It also met a core need of the central organization: without local input and control, Sugar Labs could not ensure that Sugar remained relevant and up-to-date for the needs of specific populations and, ultimately, that it had an impact. The interdependence of local and global efforts was the foundation for achieving real progress for Sugar. At the same time, it created an interesting tension as it diminished dependency on Sugar Labs central, potentially undermining the ability of the central organization to sustain itself.
- *A shared vision is essential, but visions evolve.* Many social enterprises are started by a small group of friends or colleagues who have a shared vision for how to change the world. Ensuring that this shared positive intention is translated into a clearly articulated vision for what the organization itself will focus on and how it will drive impact is vital. In the case of OLPC, profound differences of emphasis between the founders in particular, as well as other members of the founding OLPC team, ultimately drove the spin out of Sugar Labs and the eventual split of OLPC into the foundation and the association. It is not unusual for irreconcilable differences to form between the founders of a social venture regarding the implementation of their vision. Nonetheless, as OLPC demonstrates, even after a split, it is possible to find ways to continue to work toward the

original shared vision from different, complementary vantage points. Setting up a social enterprise to be a learning organization that can adapt to change is vital.

- *Drive global change while allowing for local innovation.* A small team with big ambitions cannot change the world by itself, but it can create a powerful foundation for change. Designing for viral adoption is fundamental to achieving scale while maintaining low overhead. By making intentions and outcomes clear, a disparate and distributed community can develop its own means toward a common goal. Any effort at fostering social change on a global scale must ensure clear articulation of those standards and best practices that will ensure consistency around those elements of a program that drive change. At the same time, room must be allotted for customization on the local or regional level. Many social-change organizations struggle to find the right balance between centralization and diffusion; in some cases, the central organization stifles innovation and contextualization, while in other cases lack of central control results in uneven quality and results that undermine the organization’s brand and reputation.

i “Nicholas Negroponte on One Lapop per Child,” TED video. 17:41, February 2006, http://www.ted.com/talks/nicholas_negroponte_on_one_laptop_per_child.html.

ii John Dewey, “My Pedagogic Creed,” *School Journal* 54 (January 1897): 77-80.

iii David Kolb builds on the work of Kurt Lewin to describe a learning process that starts with concrete experience followed by personal reflection on that experience. For older students and adults, the cycle continues into abstract conceptualization and active experimentation. David A. Kolb, *Experiential Learning: Experience as the Source of Learning and Development* (Englewood Cliffs, NJ: Prentice Hall, 1984).

iv Evangeline Stefanakis, *Multiple Intelligences and Portfolios: A Window into the Learner’s Mind* (Portsmoth, NH: Heinemann, 2002).

v Rebecca Herold, *Managing an Information Security and Privacy Awareness and Training Program (New York: CRC Press, 2005), 101.*

vi Cynthia Solomon, *Computer Environments for Children: A Reflection on Theories of Learning and Education* (Cambridge, MA: MIT Press, 1986).

vii David Cavallo, “Technological Fluency and the Art of Motorcycle Maintenance: Emergent Design of Learning Environments,” (PhD thesis, Massachusetts Institute of Technology [MIT] Media Lab 2000).

viii In supply-chain management, the term “upstream” means “closer to the point of production than to the point of sale.” In software development, “upstream” takes on a slightly more nuanced meaning. Software that is “upstream” is used as building blocks for software that is farther “downstream.” In the case of Sugar, the software that is immediately upstream from us is the Gnome toolkit, which we use to create the Sugar interface.