By Geert Lovink

“Software is a heap of trouble.” Scott Rosenberg

Scott Rosenberg has written an excellent book on software and open source culture called “Dreaming in Code: Two Dozen Programmers, Three Years, 4,732 Bugs, and one Quest for Transcendent Software”. In it he writes: “Our civilization runs on software. Yet the art of creating software continues to be a dark history, even to the experts. Never in history have we depended so completely on a product that so few know how to make well.” West coast IT journalist and Salon.com cofounder Scott Rosenberg produced a very readable study on the internal dynamics of the Chandler open source calendar project. Chandler was supposed to “grow up into a powerful ‘personal information manager’ for organizing and sharing calendars, emails and to-do lists.” It would be cross-platform and open source, officially realized by the Open Source Applications Foundation.

The book is chocking and boring at once. Many would recognize themselves in the stories about ‘slippage’, the delays and struggles to gain conceptual clarity amongst the drifting team members. Software is never ready and mal-functionality rules. Rosenberg spent three years as a member of the software developers team, financed and led by Lotus 1-2-3 creator and EFF cofounder Mitch Kapor, designing the Chandler calendar application that was meant to challenge the proprietary market leader Microsoft Outlook. At some point the reader gets lost in the level of detail but exactly at that moment the book takes a turn and puts the collective frustration in a wider historical perspective. Because I really enjoyed the non-academic style of this interesting contribution to the emerging field of software studies, I decided to contact Scott Rosenberg and see if he wanted to do an email interview. In between his work on his new book on the history of blogging, Scott sent me back his answers.

GL: Just to bring everyone up to date. A while ago the 1.0 version of Chandler has been released. This must have been a milestone. In your PostScript to the paperback edition from September 2007 you expressed mixed feelings about the entire project. You wrote: “For now, Google Calendar does the job for me.” You mustn’t be the only one. Before having read your book I hadn’t heard from Chandler. With 1.0 release, would that change?

SR: Chandler 1.0 is a pretty interesting piece of software in its own right —

between the "preview edition" that was where the project was when I wrote the paperback epilogue and 1.0, the developers did a lot more work, and began to flesh out parts of the program beyond the calendar. It's definitely worth a look. I'm not using it now because I'm in the middle of a big book project and don't have much free time. But when I'm done with that project I definitely want to spend more time with it.

As far as getting more attention, Chandler faces the same problem as any project that started out with really grand — and loudly proclaimed — ambitions; whatever it does achieve now is overshadowed by the original high hopes and later dashed expectations. So it will be very hard for Chandler to be "heard from," at least in the media, unless they can grow a base of users slowly and organically over the next year or two. Which is what they're trying to do, I think.

GL: Maybe I missed something. Could you explain us what people need a collaborative (open source) calendar for? What is larger culture of use of such calendar software? Mitch Kapor must have understood something about that in the early-mid 1980s when he ran his successful Lotus 1-2-3 business.

SR: I think there are at least two markets for this sort of thing. The first is pretty obvious: there are tons of companies with small workgroups (or relatively small, up to a couple of dozen people or so, I'd imagine) that need to schedule together and share stuff together. They were served poorly by Outlook (which also didn't work on Macs or Linux systems), and Outlook cost too much for them, and Kapor hoped to serve them.

The other market, which is more diffuse but in some ways bigger, is just for any smaller group of two, three, four people — a family, a couple of cofounders of a small company, a professor and her/his teaching assistants, and so on. One use case that kept coming up at OSAF was the simple one Kapor faced, as a busy guy who had a personal assistant who needed to be able to access and edit the calendar, and who also wanted to share it with his spouse.

It's important to remember, too, that Chandler didn't start out as a calendar — it began as a much more ambitious project for organizing personal information and sharing all sorts of stuff. The calendar was what emerged when the ambitions had to be scaled back.

Don’t forget, too, that when OSAF started out there was no Google Calendar, no Basecamp, pretty much nothing like we have in the way of web-based scheduling today.

GL: In the book you have carefully avoided to speak about Web 2.0. You haven’t even mentioned the term once. Still, in your analysis of what went wrong with Chandler, the choice between the operating system style calendar and the lean Web application is being played as the central dilemma of the Chandler development community. How do you judge the countless failed Web 2.0 applications that were either bought up and killed by one of the big players or that are still out there, waiting for users and a buyer?

SR: I’ve always been ambivalent about the term “Web 2.0” — indeed the people who created the term were, also, at first. It’s an ungainly label for an important phenomenon. Before it was coined, we often called it “web services” or “web apps,” and what happened was that between 2002 and 2004 or so a set of common features coalesced, and then the label “Web 2.0” just became a really convenient handle to refer to the whole bundle of phenomena. (This is the same thing that happens in programming when you add a layer of abstraction!)

So I didn’t use the term in the book, but I did write about the phenomenon, which I watched evolve out of the earliest exemplary applications — Flickr, Delicious, et al.

I’m not sure what exactly you mean by the “countless failed Web 2.0 applications” — how do you define failure in this context? The entire Silicon Valley economy is built around the notion of R&D being done not inhouse by big companies but externally by little startups. Most of them flop, some end up being acquired by bigger companies who cannibalize their code and/or their teams, and a tiny few grow into big companies in their own right. That’s the way the Valley has always worked; it’s just doing so on a bigger scale this time around because starting companies is so cheap. So the existence of all these “failures” may look bad, and of course for people who worked on them it can be painful; but it is pretty much how most of the people involved expect the whole system to work. Now, whether you think this is a good model for an industry or not is a different question; but the mere presence of all these little failures isn’t in itself viewed by people in the Valley as a sign of some larger failure. Quite the contrary.

GL: Should we, in retrospect, judge Chandler as a historic one off, or should we rather see it as ordinary collaborative open source project, which had its ups and downs? Is it fair to compare it to, let's say, Modzilla, Linux or Debian?

SR: I don't think anyone involved would consider Chandler as a typical open source project. It really was and is unique, for all sorts of reasons the book chronicles (Kapor's involvement and money, the way the team functioned more like a small startup company than a distributed open source project, the hugely ambitious design agenda and effort to put design on par with engineering, etc.). But also, as Andy Hertzfeld kept reminding me, "Every software project is unique." There really isn't a "typical" open source project.

If you look at Mozilla's history and compare it to Linux's or Apache's, they all have had hugely different stories shaped by their origins and goals and the people who got involved. And each of those stories is fascinating in its own right. Chandler is just the one I ended up telling. Even those little Web 2.0 companies that are "out there waiting for users or a buyer," they're like books that haven't yet found an audience, and maybe never will — each one of those has some sort of creative story behind it, the dreams of the people who built it and their conflicts and excitement and despair and little triumphs.

GL: If we follow Nicolas Carr in his latest book The Big Switch, the IT industry is moving to cloud computing. The implication of this is an unprecedented centralization of software, and, if you wish a further step away from the PC dominance. Chandler doesn't quite fit into that picture and tries to uphold the central role the (partly) offline, stand alone PC is still playing in the collective imaginary of the Silicon Valley engineers. Google is playing a pivotal role in this process. Should we speak of a paradigmatic clash here?

SR: This whole conflict has been playing out in the industry for decades, between centralization and decentralization. Kapor had his start in the early days of the PC, which was one of the huge pendulum swings toward decentralization, as people put their CPUs on their desks and didn't have to work through the guys (they were usually guys) in the mainframe (or minicomputer) back room. So one view is to say, cloud computing is real, it's big, but it's a pendulum swing, and eventually, as people become unhappy with its down sides, and as technology keeps changing, the pendulum will swing back in some way we can't anticipate. The other view is to say, no, this is a permanent "paradigmatic" change, a Kuhnian paradigm shift. I'm always reluctant to proclaim that, because it seems to me those changes can't even be seen until a couple of generations have passed. So if I had to I'd bet on the pendulum. But we really don't know.

GL: Most IT books you can buy are propped-up business show cases that only talk about success. Dreaming in Code is so radically different in this respect. The project drags on, and at times, the text is amazing honest, up to the point of straight out European negativity. How did you manage to do this? You wrote the book in San Francisco, not in Berlin.

SR: I'll take this as a compliment I started my career as a theater critic. I prize honesty. I can't imagine working on a book for 3-4 years if I didn't set out to be honest. When I hear "how did you manage it?" it sort of sounds to me like, "how did you get away with it?" But in fact my publisher and editor were always behind the project. My book proposal was really clear about what kind of book it was going to be.

I guess it helps that I have a record as a writer on technology going back to the early '90s, and it's all out there on the Web, and I've been writing professionally since the early '80s. I've never been a booster or a promoter, and I'm not a knee-jerk negativist, either. I like to look into new things that I don't fully understand and see if I can explain them well enough to myself that a reader will be interested, and end up understanding, too. It's the same approach I'm taking with my next book, which tries to trace the story of blogging from its beginning nearly 15 years ago to the present.

Some readers were disappointed that Dreaming in Code didn't give them more bullet points about how to improve their development projects. I had hoped that it was clear from the first page that this just wasn't going to be that kind of book. If I knew how to solve these problems I'd be busy solving them, not writing about them! But writing about them has value, nonetheless, I hope. Just serving witness to the incredibly difficult and uniquely problematic work that software developers do — that was my aim, in the end.

GL: At two-third of the book, you leave the inner dynamics of the stagnating project and turn to history of software to show that it's always been like this. In the chapter on Engineers and Artists you show a lot of the historical continuity between now and forty years ago. There are efforts under way to build up Software Studies from a critical, humanities perspective. Would you mind to be called a software anthropologist? Software rules the world, but the research into its modes of becoming is rare. What directions would you like something like software studies to take?

SR: I'd be happy to be called a software anthropologist — in some ways that's exactly the approach I took — with the caveat that my anthropology "degree" is strictly honorary. I actually studied history and literature and can't claim too deep a knowledge of anthropology. And it's been nearly 30 years since I spent much time on campus so it's hard for me to know what the best route for pursuing a sort of humanistic software studies field would be. There are fascinating perspectives coming in from the law schools, and analogies from biology, there are sociological and philosophical lines of inquiry — there's tons of work that could be done once we accept that software, and the people who make it, are worthy of study. I'd hope that that is an uncontroversial attitude by now, but I'm not sure it is.

—

Scott Rosenberg, Dreaming in Code, Three Rivers Press, New York, 2008
Scott's blog: http://www.wordyard.com/
The book's website: http://www.dreamingincode.com/