The Inmates Are Running The Asylum
Why High-Tech Products Drive Us Crazy And How To Restore The Sanity

By Alan Cooper
Reviewed by Tim Renczes

In traditional software development environments, organizational hierarchies have revolved around the C.E.O or president of the company in a top-down approach. Decisions are made at higher levels, and subsequently filtered down to Product Line Managers, Project Managers, and lastly, Software Engineers and Programmers. This is not necessarily the case – in fact, the high-tech industry has inadvertently put programmers and engineers in charge. Programmers and engineers run the show and end up being "back-seat drivers" for product development. This is the essence of Alan Cooper's book, "The Inmates Are Running The Asylum".

Software engineers try very hard to make their software easy to use. They believe that their products are as easy to use as is *technically* possible. As engineers, their belief is in technology. They see power and flexibility, richness of features and functionality in their software. What they don't see is how difficult their software is to use for people who use it on a daily basis. As such, it is hard for them to see clearly the nature of the problems plaguing the software industry. The result is what Cooper calls "Dancing Bearware". If one saw a dancing bear on the street, we would be amazed not because the bear dances well, but that the bear dances at all. Cooper likens this to software today in that people will overlook interaction problems because the software performs its function, albeit problematically and inefficiently.

"Featuritis" seems to be the dominant problem, with most companies releasing new versions of their software on an annual basis that incorporate an ever-increasing number of features. Software manufacturers perceive adding new features to a product as being "free", so they gratuitously devote resources to feature development so long as the product ships on time. In fact, as Alan Cooper observes: "Most product managers would rather ship a failure on time than risk going late".

The root of the problem lies in the software creation process. When a software project is laid out, programmers begin programming right away. Design only takes place when the product is finished to make the finished product easier to use. The flaw with this approach is that programmers design for themselves, not the user. They make design decisions at every step of the process. The programmer "decides how each procedure will call each other procedure, how information and status will be shared, stored and changed, and how its validity will be guaranteed". Each decision and the success of each one depend on the programmer's ability to bring their experience and judgment to bear. The way a program is structured directly affects the usability of the end product, and ultimately, a company's revenue.

When a project's goals are initially scoped, it is typical to develop a feature list. Project members will spend hours, maybe days determining what features should be incorporated into the final product. Otherwise known as "feature list bargaining", features are traded-off against the project's deadline(s). What managers don't realize is that programmers have the upper hand for determining what features will actually be implemented. Programmers establish how long it will take to implement each item on the list, so they can force items to the bottom of the list merely by overestimating the time necessary to complete them. In this way, programmers manage to thwart the traditional decision-making process, and as such, programmers are completely in control in a bottom-up fashion, rather than the top-down approach. As the title of the book suggests, "The Inmates (Programmers) are running the Asylum (The Business)".

The consequences of this approach of software design are far-reaching and result in bad products. One example of a quote from a C.E.O in the book is "Don't throw out the prototype. Let's use it as the foundation for the real product". I've seen firsthand the effects of this approach. Managers and programmers alike find it very difficult to throw away existing work that has been done on a product. They liken it to throwing money out the window. The reality is that prototypes are initial product concepts – they are put together under extreme time constraints to showcase the viability of a product. As Alan Cooper stresses in the book: "It is essential to have design completed before coding begins". With a prototype, there is no time to do initial design work. Interactive products need to be designed by interaction designers instead of by software engineers. When prototyping, interaction designers are rarely used, which is why prototypes should never be used as the base upon which a product will be developed. The cost benefits of doing design prior to development are astronomical. Cooper compares software development to the movie industry whereby it can cost millions of dollars to re-shoot a movie scene if not planned out properly to begin with.

In the book, Cooper introduces us to the very important notion of Goal-Directed design. He strongly suggests designing for only one particular type of person so as to improve a product's chances of succeeding. Cooper suggests targeting 10% of the market and working to make them 100% ecstatic about the product. Cooper uses a very interesting example with Chrysler Corporation. When Robert Lutz, chairman of Chrysler, was considering developing the new Dodge Ram pickup, 80% of people in focus groups hated it; however, Mr. Lutz went ahead with production because the remaining 20% *loved* it. The Dodge Ram went on to become a very profitable vehicle for Chrysler. This example proved that having people love your product, even if it is only a minority, is how you succeed. This is particularly true in the IT industry, whereby targeting a vertical market is necessary in order to be profitable.

When targeting a particular customer segment, Cooper suggests the use of personas during the design process so that design decisions can be made with a particular persona in mind. Cooper goes so far as to insist that a name be given to a persona and to refer to the persona by name, rather than as "the user". When designing a product, every person on the design team will have different ideas of what "the user" needs or wants. As such, every person will make

individual decisions based off these perceptions. This ultimately results in a poorly articulated product that doesn't meet "the user's" needs. If a persona is narrowly defined such as: "Joe is a 35 year old lawyer who drives a Porsche. He owns a cell phone, pager, Blackberry and always is on a tight schedule", it is easy to relate to Joe's needs, namely time-savings, and the need for information at his fingertips. Joe is more likely to get a product that's designed for him.

Sometimes one persona isn't sufficient to cover all the users of a product. In these cases, Cooper suggests the development of multiple personas, to a maximum of three. If more than three personas are created, Cooper believes that the problem is too large and must be broken down further. As these personas are developed, a cast of characters is formed. When these characters are precisely defined, the cast of characters becomes a design taxonomy which has great power to explain our design decisions. As in Joe's case, if we added a wireless feature to the product, its justification would lie in the fact that Joe is rarely in the office rather than an explanation of "The user *might* be out of the office and *could* use this". This approach takes the guessing out of the design decision.

Cooper takes the notion of personas further and introduces the use of scenarios. By creating daily use, necessary use, and edge case scenarios using personas, we can even more clearly define in what context Joe might use a particular feature that we are considering. In this way, businesses will be in a better position to perform a cost-benefit analysis for each potential new feature. By comparing potential sales for a particular persona to the development costs associated with a new feature, the business can make an informed decision as to which features will benefit its customer base most. Businesses can shift the power away from programmers to management and their client base.

Another issue touched upon by Cooper is "The Customer-Driven Death Spiral". One of the buzzwords for the twenty-first century is that a company should be "customer driven" – they should always be aware and respond to customer problems; however, this approach can be fatal when the customer is "holding the cheque". Management, marketing, sales, and product development will each have different views of how a product should be developed, but the customer ultimately is the one who pays for the product at the end of the day. Cooper stresses the difference between *listening to* and *following* a customer. While listening to a customer is good, following a customer by merely doing what a customer tells you to do is bad. As soon as a company allows customers to dictate what features a product will have, the product will lose coherency in its design. The product will mutate from one release to another, instead of growing in an orderly manner. While each customer will get the features they want, they will also have to avoid features they don't want. As a result, the product's desirability will decrease, sales will decrease, and the company will falter. The customer does not have the long-term interests of the company in mind.

Overall, this book was an incredible read and an enlightenment into the world of interaction design and the design decision methodology. Alan Cooper's extensive use of examples and case studies of companies such as Sony, Microsoft, Logitech, etc... lend proof to every concept he discussed. He always

made comparisons between what current design standards are, and what should be done to improve upon them. Alan Cooper's consulting experience in the field of interaction design also was of interest as he provided hands-on examples of the application of his methodology. By adopting company-wide standards for design, companies can begin to benefit and reap the rewards of creating user-friendly products. The challenge now is to create decision support systems that complement the design process and incorporate Alan Cooper's learning.