

IDC

MODIFICATION NOTICE

The following report has been modified from its original version as follows:

Content has been changed

all references to ai*soft have been changed to Authorgenics.

(ai*soft changed its name to Authorgenics in June of 1997)

Text formatting has been changed

by the addition of paragraph breaks, page breaks and enlargement of the type face.

INTELLECTUAL PROPERTY NOTICE

The original text was written by IDC (International Data Corp.). The notices in this document do not intend to claim any rights to the part of this work that is intellectual property of IDC.

IDC Review of Authorgenics inc's product technology

Introduction

IDC visited Authorgenics, inc. for one day in February 1997 to review the company's product technology. This review took the form a day-long presentation about the technology foundation which Authorgenics has created, and included limited exposure to running applications that were created with various instantiations of this foundation that could be thought of as development environments.

This report attempts to describe what we perceive is the nature of Authorgenics inc.'s product technology, provide a hint of how it works, and suggest various ways in which it can be positioned in the context of today's software markets. Given the time constraint we could only make a limited attempt to "validate" that what was described to us actually existed in all its potential facets, and that the applications we were shown were developed exclusively through the use of Authorgenics technology. Our task was primarily to understand what Authorgenics has (in the context of a one day meeting) and comment on the market positioning of some of its instantiations.

Background

During the middle of this century, when the technology that formed the foundation for the modern computer industry was emerging, there were two competing computation models. One was represented by the von Neumann machine, a symbolic processor that executed a single instruction at a time. The other was represented initially by a device called the Perceptron, and was focused on the task of perception.

In a practical and commercial context, the former quickly won out over the latter, and went on to dominate the computer industry. Virtually all computers today, from PCs to mainframes, run on the von Neumann architecture and do symbolic processing.

In recent years, the other computational model has been making a quiet comeback, primarily in the context of pattern recognition. For example, the military has been using this computational model in target acquisition systems for missiles which autonomously recognized various potential targets (tanks, buildings, groups of people) during its final 2 minutes of flight.

Computers (of the von Neumann type) have advanced in price and performance to the point where today they can actually be used to simulate (in software) some of the aspects of the perceptual computational model in a cost effective manner. This was evidenced earlier in this decade in software products to create and train neural networks for perception tasks. These were able to perform discrimination tasks extremely well and therefore recognize patterns, for example in data, that made other techniques pale by comparison. One emerging manifestation of this today is an aspect of data warehousing called data mining, in which the data mining tools can "discover" patterns in the data not previously perceived. Anyone who has looked into this technology quickly learns that it is both powerful and difficult to comprehend.

The human brain, on a much more massively complex scale, operates in a fashion that uses both computational models in an integrated fashion. One complex aspect of behavior that this synergistic amalgam of computational models enables is the joint process of discovery and learning. This, of course, is a cumulative process that combines prior knowledge/experience and new stimulus guided by some mechanism that introduces a sense of relevance. It also allows us to create new knowledge, or in

the case of an artist, more tangible artifacts. For each of us this is a continuous , life-long process. It means that we have the potential to do anything; we just need the proper training to acquire the necessary skills and knowledge.

Authorgenics

Authorgenics inc. has created a "body" of software that employs both computational models discussed above. The result is an extensible creation engine. Its current instantiations are primarily focused on the task of "creating" software source code and related natural language text for specific applications domains to which it has been applied.

Given the general purpose nature of the foundation technology, the Authorgenics technology, which we will refer to as Author, it may have broader uses than this, but these were not discussed. Since Author takes advantage of accumulated knowledge and skills, it can very rapidly create the software application and text from a relatively minimal set of application specific information. Hence the positioning of these instantiations of Author in the context of Very Rapid Application Development or VRAD. It is the rapidity with which applications or (products) and their related documentation can be created, that differentiates Author from other ways (combinations of methods and tools) of creating comparable software.

Another way of categorizing Author is that it will participate in an emerging market for intelligent integrated development environments, IIDES. Rather than trying to provide "point solution" tools that automate one phase of the development life cycle, IIDES are imbued with knowledge about software architectures, programming conventions, etc. and driven by sophisticated rule-based engines that will combine this general development knowledge with application specific requirements to create whole programs.

There are three different areas of research that is ongoing in this area and they are orthogonal to each other.

• Rules-based systems that take advantage of pattern matching

• Learning systems that use techniques like genetic algorithms to combine existing software development knowledge in innovative ways to meet new requirements

• Automated Assistants based on the work of Charles Rich (MIT AI Lab) and others that processes clichés and patterns, software architectures and design patterns to produce whole architectures

While Authorgenics uses many of the words common to the artificial intelligence community to describe its technology foundation, they do so with different semantics. This complicates the task of fully understanding the full "power" of Author.

As best we can tell, Author is in the first category - rules-based systems that take advantage of pattern matching. We believe terms like "learning" and "discovery" may overstate the current state of Author's technology. Most rule based engines use pattern matching to find potential rules within the underlying knowledge base and also pattern match against attributes of objects in domain models. While this may seem like learning and discovery, at least within the research community, these are viewed as being different.

Primary Characteristics

Caveat

It is difficult in one day to acquire an in-depth understanding of what IDC perceives to be a significant departure from the usual parade of new software products and application development tools and environments to which we are exposed each month. Thus, despite the heroic efforts of Brian Stack, Author's primary creator, to pack several days of training into one day, we can only present our perceptions based on an imperfect understanding.

We believe the old adage "things are the way they are because they got that way," applies to Author in its current state of evolution. Author is the result of what it has "grown" to be over a period of years, secreting capabilities over time.

Perceived strengths

The following are the main strengths which we perceive to be apparent in Author as it was presented to us. In the comments that follow we use the term "application" to denote both end user applications and products, i.e., something that could be licensed or sold to another. In the latter case, Author's involvement would probably be transparent to the ultimate product user.

Ÿ Author has demonstrated an ability to "create" at least one each of a variety of application types.

- ž data-oriented transaction processing, primarily in the context of "tabular" data.
- ž process control, including event driven processes
- ž communications, i.e., data communications
- ž robotics, specifically equipment performing complicated tasks involving test tubes

Ÿ Author has demonstrated an ability to perform very rapid application creation, even when addressing a "new" application domain. The evidence for this is somewhat anecdotal and based on a limited set of examples so far. Never-the-less, the examples given seem to be quite remarkable and suggest "orders of magnitude" improvements in development productivity over prior practices.

Ÿ Author has demonstrated the ability to capture a high level (of abstraction) specification of application process definition and create working applications with this minimal input. These working applications are generated quickly that some people might be tempted to think of the resulting applications as prototypes, but they are in fact working applications, not prototypes. The distinction is important.

Developers prototype in order to model or analyze some aspect (performance, scalability, user interface) of the eventual application. Once this analysis has been accomplished the prototype is discarded. This is not the case with Author which generates a working application every time.

Ÿ Author has demonstrated multiple modes of "creation" or authoring

- ž software source code
- ž database schema
- ž specification documentation (complete and detailed)
- ž user guides
- ž help files (field and program context-sensitive)

Ÿ Author has an extensible knowledge-base (we use this term loosely) which suggests the ability to incorporate support for additional:
software languages

programming styles
software architectures and data models
natural languages interfaces to external software environments, e.g., specific databases, graphical user interfaces

Ÿ Author seems to have the potential foundation for the support of creative processes for more than just software-based applications, e.g., mechanical design, manufacturing process definition, etc. Again, this perception is based on our limited exposure to Author. To perform such tasks, additional knowledge would have to be added to Author.

Current status and other remarkable characteristics

Author has limited support for the more "popular" languages in use today, other than Business Basic. Depending on where Author is positioned in the overall market for application development environments, additional languages become important, e.g., C, C++, Java or HTML. Preliminary trials with Author in this regard have already begun.

Ÿ Although not limited exclusively to SCO Unix, it appears that the majority of Author's platform experience focused has been focused there.

Ÿ Author itself written in Business Basic. Its conversion to another language, if deemed necessary, could probably be done by Author itself (once it was capable of "creating" in that language, e.g., C++).

Ÿ Author's user interface is character-based and its support for creating graphical user interfaces (GUIs) has only been demonstrated in prototypes and exploratory evaluations. The need for creating GUIs for applications is "market driven", i.e., it depends on the market segments in which Author is positioned.

Ÿ Similar comments apply to Author's "creation" support for concepts we perceived to be limited in its current instantiations. We are referring to things like object-oriented design concepts, use of abstract datatypes, knowledge of complex and full class relationships. There were hints of some of these, but we were not capable of fully appreciating what may in fact be there.

Ÿ In the same vein, it was unclear what provisions existed conceptually for the capture of rules related to an application domain. Attempts to discuss this just lead to frustration for both presenters and ourselves.

Ÿ The scalability of created products or applications has only been partially demonstrated. This may change shortly given some of Author's more recent prospects. Similarly, the ease with which Author will integrate with, or create applications that integrate with, existing (sometimes called legacy systems) systems, or how well it will operate in heterogeneous, distributed systems, has only been partially demonstrated.

Challenges/Opportunities

In what follows, there is no particular order of importance.

Striking a balance between universal potential and specific focus

Author has a foundation that includes (apparently) a significant potential to be extended eventually to

perform a wide variety of creative tasks. Ultimately perhaps it could be all things to all people. But as Lord Keynes said, "in the long run we're all dead." So some choices need to be made to narrow the field of opportunities to be pursued.

The positioning being discussed is as a development tool for software applications, including for the creation of other vendors' products. We believe this type of focus is appropriate. We would encourage even sharper focus, perhaps within a selected subset of the application domains already demonstrated with Author.

These could be expanded over time, but an early focus should allow Authorgenics to establish a dominant market position progressively in each specific market segment. Given the nature of the creation environment we saw, we would position Author in one of the application development tool sub-markets like analysis and design tools or RAD (rapid application development). Some of our other comments are in this market context.

Paradigm Shift

Author seems to represent something very unique. As such, it is a distinct innovation in the world of application development environments. Some people would characterize it as a revolution.

It implies a paradigm shift for developers, i.e., it requires developers to adopt a different perspective with regard to application development. Products and technology that represent paradigm shifts encounter an inertia that limits their adoption, forcing it to be adopted progressively over time. This gradual adoption is typically represented by a technology adoption curve with identifiable stages, which had been expertly described by the popular author Geoffrey Moore in his two books "Crossing the Chasm" and "Inside the Tornado." First comes the "innovators", quickly followed by the "early adopters." The next group are called the "early majority."

Author will do well with the early adopters. These people are willing to take a risk (e.g., on a new technology or on the products of a small company) to achieve an order of magnitude improvement in some important measure such as development productivity or time to market. Author has the potential for such benefits and will be able to take advantage of reference accounts that validate the magnitude of the benefit and a sales approach that emphasizes an actual demonstration of the power of Author on a prospect's application.

We believe Author faces a larger challenge moving into the broader "early majority." People here are pragmatists, that look for validation within their peer group (not the early adopters) that the new technology works and is mature enough so as to represent a negligible risk should they attempt to use it.

The magnitude of benefit, impressive to the early adopters, will have less credence with the early majority. They have been burnt too many times by the latest and greatest innovation. We believe Authorgenics will need a significant infusion of marketing talent to make the transition into the early majority.

Author is very unique

That's good, right? Yes, to a point. However, it may be too unique for the typical risk adverse prospect. Because of the unique amalgam of the computational engine in Author's foundation technology, it's extremely difficult for a typical developer or his manager to understand "how" Author

does what it does, and the scope of Author's creative capabilities, specifically any claims that Author can support the particular types of products or applications for which they would use it.

This challenge may also exist in the context of Author's use by other product developers to create products. For example, as a product creator, the very unique nature of Author does not lend itself to easy audit. For example, let's assume some time in the future the Federal Drug Administration is tasked by Congress to approve medical equipment and that Author has been used to create the software for such a equipment. e.g., a blood analyzer. The FDA might, as part of the certification process, want to understand how the software controlling this medical device was created in order to prove that the creation process got from the equipment specification to executable software in a predictable and repeatable fashion. Although this scenario is hypothetical, this is a possibility, if not for the FDA, then perhaps NASA or the FAA. Author does not lend itself easily to such an audit.

We feel this is an inhibitor to adoption. We understand it will be countered by demonstrations of order-of-magnitude improvements in development productivity, either directly with the prospect or indirectly validated with reference accounts. Which leads us to our next point.

Metric-based comparisons of Author with alternate modes of development

A key differentiator for Author is the rapidity with which Author can create an application or a product and therefore provide the attendant productivity benefits to the developing organization. Given the importance of this, we believe Authorgenics should augment its use of reference accounts and demonstrations to promote these claims.

Specifically, we recommend an approach based on function-point analysis. Software Productivity Research, Inc. (Burlington, MA) maintains a set of measures, e.g., developer person-months per function point, for a wide variety of languages and tools (all kinds of tools) used for application development. They provide a very credible benchmark for productivity claims for new tools and technology. Their reputation is established. They could quickly establish quantitative comparisons of Author against its nearest competitors, analysis and design tools and RAD tools.

Recognizing Market Momentum

We believe there is a significant difference between the development approach that Author represents (is) and the current momentum of the primary target markets for Author. Author will have to counter these trends.

Two we can cite specifically are efforts by Rational Software and its allies to promote the Unified Method Language (and indirectly, the associated methodology) as a standard within the Object Management Group (OMG) and elsewhere.

The second trend is the rapidly emerging emphasis on component software. This means that Authorgenics will have to simultaneously introduce itself, gain attention and try to lure potential users away from a direction for application development which users are being told is the trend. That is, Authorgenics will be promoting a message that basically says "the vast majority are marching in the wrong direction." And it must first gain their attention. This is a time consuming and resource intensive activity.

If the computer industry market place has taught us anything, it is that the best technical solution frequently is thwarted by a competitor with strong marketing and deep pockets. Authorgenics would do well to compete in markets being ignored by the major market leaders like Rational or, where appropriate, align itself in a way that is complementary to their marketing and sales strategies.

We believe Authorgenics understands this and intends to target markets that are "tool hungry" in the sense that application developers in those markets either have no tools or only a few point solutions available for application development.

Authorgenics has an opportunity to capture a dominant share of these emerging markets, e.g., personal appliances, embedded systems, and other non-traditional systems. Since many of these newer devices and applications will eventually need to be integrated, Authorgenics would then be in an excellent position to capture additional business in that context, as well.

Conclusion

Authorgenics and Author are very unique and, as such, represent a paradigm shift in application development. The product, as constituted today, needs to be upgraded with support for GUIs and more mainstream languages.

Since it is extensible, it has potential for a variety of potential markets, not just in the context of software development. Even within the software development tools markets (CASE or RAD), Authorgenics would do well to focus on areas where it can achieve a dominant market share.