EGL

Epsilon Strategy

March 27, 1990

1. Introduction

An appreciable portion of our development resources are currently engaged in creating Epsilon to run on the IBM RS/6000. Shortly, this effort will be matched proportionately by increased effort in other parts of the organization. By this time next year, the resources being applied to Epsilon by Rational home-office and field, and customer personnel will probably exceed development resources.

Clearly, Epsilon will play a major role in Rational's future. This document is an attempt to place it in the context of the history and future of Rational and its products.

2. Why Epsilon?

Around the time that Beta began shipping, there was a realization that there were significant product deficiencies. The project that was started at that time was called Zeta (leaving room for Gamma, Delta and Epsilon). By the time this was reduced to practice, it had become Epsilon and the sights had shifted some, but the basic goals were still in place. Some of the Zeta goals, most notably CMVC, were incorporated into Delta.

The original R1000 Epsilon project had as its goal implementing technical fixes to Delta deficiencies. The current AIX Epsilon project is intended to, in addition, address market weaknesses of the current product.

2.1. Delta Deficiencies

The name *Delta* is used to exemplify the problems in the current product, most of which are inherited from its predecessors. R1000 Epsilon was an effort to fix the technical deficiencies of the product as it existed in Delta. The current Epsilon effort is very similar in its technical characteristics, but addresses other, more market-related, product problems. Epsilon (and this section) concentrates on the central system aspects of the product, not on extensions into other lifecycle or product areas, additional language support, etc.

Delta is a very useful product and a significant player in the bulk of our revenue for the next few years. The following enumeration of product weaknesses is not balanced; it focuses entirely on the problems.

2.1.1. Operations

Backup and recovery are inadequate to the need and don't work all that well anyway. It is only useful for catastrophic recovery, and has a high failure rate in that role. Combined with the software assumption of perfect disks, industry-acceptable disk error rates become system-killers.

The generation of disk garbage greatly decreases effective disk utilization and, as commonly encountered, introduces high-load computation at times of peak system utilization. In addition, diagnostic tools for disk usage are slow, unwieldy, and generate garbage themselves.

Snapshots and, to a lesser degree, the other daemons also complicate system maintenance and interfere with normal usage of the system. Snapshots also make it difficult for users to estimate the magnitude of loss due to system crashes.

Delta R1000's are prone to a variety of system crashes, hangs, and unconstrained resource consumptions that exacerbate all of the above. Most of these problems require either a debugger or a crash dump to even explain what went wrong, much less to initiate the process of providing a fix. Debuggers are often not allowed, and when they are require hours of setup and usage to deal with the simplest problem. Crash dumps are more convenient, but are seldom useful.

Facilities are deficient for controlling machine usage resource consumption, for tuning system performance to function best in a particular environment, and for predicting future needs and problems.

The only *operator* interfaces are either the general Ada interfaces or command interpreters designed for internal diagnosis. As a result, a professional operator finds the R1000 deficient in both form and content.

2.1.2. Ease of Use

The Delta interface is text-oriented, complicated and different. At the very least, this involves a steep learning curve. The emphasis is always on functionality rather than presentation. Incremental compilation is an example of a feature that is immensely powerful, but difficult to become comfortable with.

Much of the functionality is presented in terms that make the implementation easier. There is confusion between invariants that programs have to maintain and invariants that users must maintain. A large number of interfaces don't have any way of dealing with unexpected corruption, whatever the cause.

The *user* interfaces of the system are modified programmatic interfaces for execution from command windows. They are inappropriate for simple user interaction and insufficient for programming. Mechanisms are presented as solutions, resulting in the delivery of neither.

Significant Delta functionality is transparently added on to an otherwise integrated system: Access Control, CMVC, and RDF are principal examples.

2.1.3. Prototype

Many of the features of Delta don't scale well to large projects, the part of the market that we can most closely call *ours*.

Since the recognition that real projects would require many machines, the primary changes to support them has been in how we presented the product; a very strong statement on the value of positioning. Delta support for multi-machine development consists of network copy and recompilation. As customers actually complete large project (vis. Bofors), they detect the difference between the positioning and the actuality.

CMVC supports an attractive notion of fully-enfoliated, compiled views, but these views are so expensive in terms of size and creation time as to make them too expensive to actually use properly.

Incremental compilation facilities work well for initial spec changes, but changes to implementations are still have batch response characteristics. Small changes on large systems are dominated by coding and loading, just as in conventional systems.

2.1.4. Universal Host Development

Incremental target compilation has few of the advantages of R1000 compilation.

The Universal Host concept hasn't been carried all the way to the target in the form of testing facilities for portions of the system that can't run on the R1000.

2.1.5. Open Systems

Although we publish a broad interface and promote the notion that we have an open system, we clearly provide functionality to internal tools, e.g. object editors, that we don't export for customer usage. Customers have shown

an excellent ability to detect these areas.

2.2. Market Weaknesses

Even when completed, R1000 Epsilon would have faced a number of market weaknesses probably more serious than the technical ones. Had it been technically feasible to solve these with a Delta-based system, a much harder decision would have resulted.

2.2.1. Price, Entry Price, and Performance

Using the 300C as a basis, the entry level R1000 is charitably a $50K/MIP machine in a market where $1K/MIP is normal and DEC is getting losing market share with $10K/MIP. At the entry level, similarly configured machines are available at a quarter the price (with much better performance). To make matters worse, we sell these systems as multi-user timesharing machines where the comparison machines are individual workstations; the typical customer needs a workstation for each user in addition.[1]

An order of magnitude difference in delivered performance represents a qualitative, not just quantitative, difference in capability. For heavily-loaded customer machines, we are conceding two orders of magnitude in performance. This will present options for competitive (or substitutive) products with significantly lower development investment than was required for our product.[2]

2.2.2. Proprietary

In the current market, proprietary hardware and operating systems are considered the province of the greedy or the stupid. We have been able to get the benefit of the doubt up until now, but increasingly we are perceived as both; the preferred market position is neither. The above comparison of our price and performance characteristics is a case study in why purchasers don't want proprietary systems.

In addition to being a drag on market acceptance, proprietary hardware and operating software is a drag on corporate resources. Historically, between a fifth and a half of the development budget has been spent on direct hardware development, microcode and tools, driver and network software, and in replication of general-purpose software required for a machine that only runs software developed in-house. The bulk of this resource could have been used to develop product characteristics to increase product differentiation and market share. In addition to confusing customers about what business we were in, we confused ourselves. Other parts of the organization also incur costs related to being in the computer business; some of these are directly offset by additional revenue and others are not.

2.2.3. Aggressively Monolinguistic

Delta only supports Ada. All programs running on Delta are written in Ada. Any user who runs commands needs to know something about Ada. Mechanisms that are provided for Ada aren't provided for other Rational-defined object types and are built in ways that don't generalize to other object types. Execution of non-Ada programs on the R1000 won't be easy and is unlikely to have adequate performance.

---

[1] It is impossible to resist pointing out that a product that can be sold under this large an apparent disadvantage must have some redeeming social value.

[2] Approaches that are silly at 0.1 MIP can be quite productive at 40 MIP.

2.2.4. Security

In the near future, C2 security certification will be required to sell systems to the government, and many applications will seek B-level security. There is some work in making Delta appear to be C2; it likely that no R1000-based system could ever be made B-level secure.

3. What is Epsilon?

The exact feature set and functionality of Epsilon is still to be determined. There is nothing even closely resembling a product plan. A number of the early directions have been pursued to the point that they will, in some form, clearly be part of any Epsilon release. Other features are likely to appear in either the first or the second Epsilon product release. Consistent with Epsilon history, there are only a few features that have been conceived of, but not included in the development effort.

3.1. Intrinsic Features

The following features are already built into Epsilon. Their inclusion is *free*, in the sense that the most efficient path to completing Epsilon from where it stands today includes them.

3.1.1. True Multi-Machine Development

Epsilon provides true multi-machine development for Ada programs. It is possible to copy objects and subsystems between machines on a shared network and it is possible to compile units in a subsystem on one machine against units on another machine.

3.1.2. Integrated CMVC

Epsilon CMVC is integrated into the basic object management layer software. This integration removes some of the *added-on* characteristics of Delta CMVC. In addition, differential storage of views makes them efficient to use, both in terms of time to create and space used.

3.1.3. Improved Compilation and Import Model

Epsilon compilation achieves the results of incremental compilation from Delta without the editor restrictions imposed there. This allows casual user and tool-invoked changes to get the same benefits as carefully crafted Delta changes.[3]

The associated improvements in the import model will allow new specs to be imported into a view that requires the changes without requiring the same update to other using subsystems. This import will also require minimal recompilation.

The compilation model for native and target compilation should be identical, removing the distinctions that exist in Delta. Epsilon compilation may provide reduced functionality in areas where the R1000 hardware was instrumental.

---

[3] Epsilon incremental compilation will be simple enough that it may be harder to demo.

3.1.4. Efficient Disk Utilization

Epsilon has no problems corresponding to the Delta garbage generation/collection problem. As a result disks can be more fully utilized without impacting storage. In addition, Diana trees (a major portion of the storage required for most users) are stored in roughly half the space of their Delta counterparts. These changes account for between double and quadruple effective disk capacity for the same contents. Combined with the reduced need for replication across machines[4] and the differential nature of views, the total disk investment for a large customer is appreciable; a factor of up to 10 for the best case seems attainable(not to be quoted to customers).

3.1.5. Improved Operability

There are no snapshots, garbage generation, or object manager daemons. The command interpreter provided at the console is capable of a wide variety of functions, including running any environment program. System startup is appreciably faster than Delta. Common commands to find out resource utilization and limits are better engineered and take seconds instead of hours.

3.1.6. Performance

R1000 Epsilon was slated to have better general performance than Delta because of improved system efficiency and higher page-reference locality. Usage of the RS/6000 should further improve this performance, especially in areas where R1000-specific mechanisms aren't being emulated. All functions are expected to be noticeably faster, though batch compilation may provide the least impressive improvement.

3.1.7. Improved User Interface

Epsilon has a number of improvements to the user interface in the form of more consistent and simpler presentation of what was in Delta and the removal of some of the more complicated interactions from Delta (e.g. incremental compilation).

3.1.8. Platform-Provided Functionality

Running on top of AIX provides a number of facilities and industry standards in a form that will require greatly reduced maintenance and improvement effort. In particular, AIX is B-level secure and provides a superset of the standard networking provided in Delta. As FDDI, OSI or whatever gain acceptance, we will be able to adapt without the need to implement the associated standards.

3.1.9. Positioning

The AIX platform immediately improves product positioning by taking us out of the computer business and into providing a CASE environment on an industry-standard platform. This is not the same as running on all industry-standard platforms, but it is very close as long as IBM continues to make RS6000 a credible platform.

3.2. Likely Features

There are a number of features that are considered likely, but may not be in the first release.

---

[4]Cross-machine references to released views may be automatically cached on other machines, requiring some space

3.2.1. AIX and NFS File Integration

The ability to reference objects through the standard Unix file mechanisms will provide extended services for integrating environment functionality with existing Unix (and other) data and applications.

3.2.2. Standard Window Interface

Epsilon is headed in the direction of providing a standard Motif (or whatever IBM adopts) user interface. This will simplify the user training and casual user interaction significantly and remove common image and selling-objection problems. Providing the facility in the initial release is viewed as entailing significant schedule risk for the first release.

3.3. Possible Features

The following have been mentioned, but haven't received enough effort to commit one way or the other.

3.3.1. Integrated Database Support

Use of SQL database interfaces to access databases through commercially available databases. Could form the basis for a more extensible work order facility.

3.3.2. Additional Language Support

Epsilon might include first-class support for other languages of interest or simply the ability to call and be called from programs in other languages.

3.3.3. Separate Site CMVC Support

Epsilon might attempt to provide CMVC support for development of programs at physically separate sites. This is also called multi-primary development; it is a natural outcome of the Bofors integration method.

4. When is Epsilon?

When Epsilon will appear is a matter of great interest and relatively little concrete information. The last schedule produced indicates that the earliest conceivable Alpha date would fall in 91Q2. Since that schedule was produced, resource diversions and IBM delays have almost certainly moved that date into 91Q3. The times between Alpha, Beta, and Production dates are not etched in stone and will depend on a number of factors: software stability, transition requirements, documentation, training, and the general business climate.

One of the areas that needs work is in defining better what Epsilon is and exactly what work must be done. Another is to make sure that Epsilon starts to get the resources it needs. These activities are required to get a quality Epsilon out of development as early as possible. Effective coordination of the entire product and all of the players throughout the organization will be essential to reducing the time from development Alpha to Production.

5. Technical Transition

The *technical transition* is the process whereby customers who have been using Delta are provided with a path to easily and effectively make the transition to Epsilon. We have to be able to make the transition to Epsilon without dropping any customers on the floor **and** without paralyzing the field organization for an extended period. If we don't bring our current customers along, we won't find replacements, much less new converts. If all of our resources are tied up dealing with our current customers, it will be hard to move forward.

Effective transition is a process, not a piece of software. Significant software will be necessary, but not sufficient. The transition process will start with customer education and appropriate interface specs for Epsilon, available for use on Delta. Tools will need to be provided that analyze the Delta contents of a customer's shop and isolate areas of difficulty, decisions that are required, and produce media (or network transactions) to be loaded onto the Epsilon machine. It is anticipated that there will be Epsilon-compatibility mode interfaces for Delta and Delta-compatibility mode interfaces to run on Epsilon. There will also be interfaces for which one or both of these compatibility modes are inappropriate or infeasible.

The first deliverables of this process will have to start appearing within the next year to begin to give everyone involved some time to adjust. Although the first deliveries will be subject to change, the Epsilon specs will have to be available earlier than has been typical for Rational releases.

6. Business Transition

A fundamental question is how the business will make the transition from a hardware-dominated revenue stream in 1990 to a software and services revenue stream in 1990.