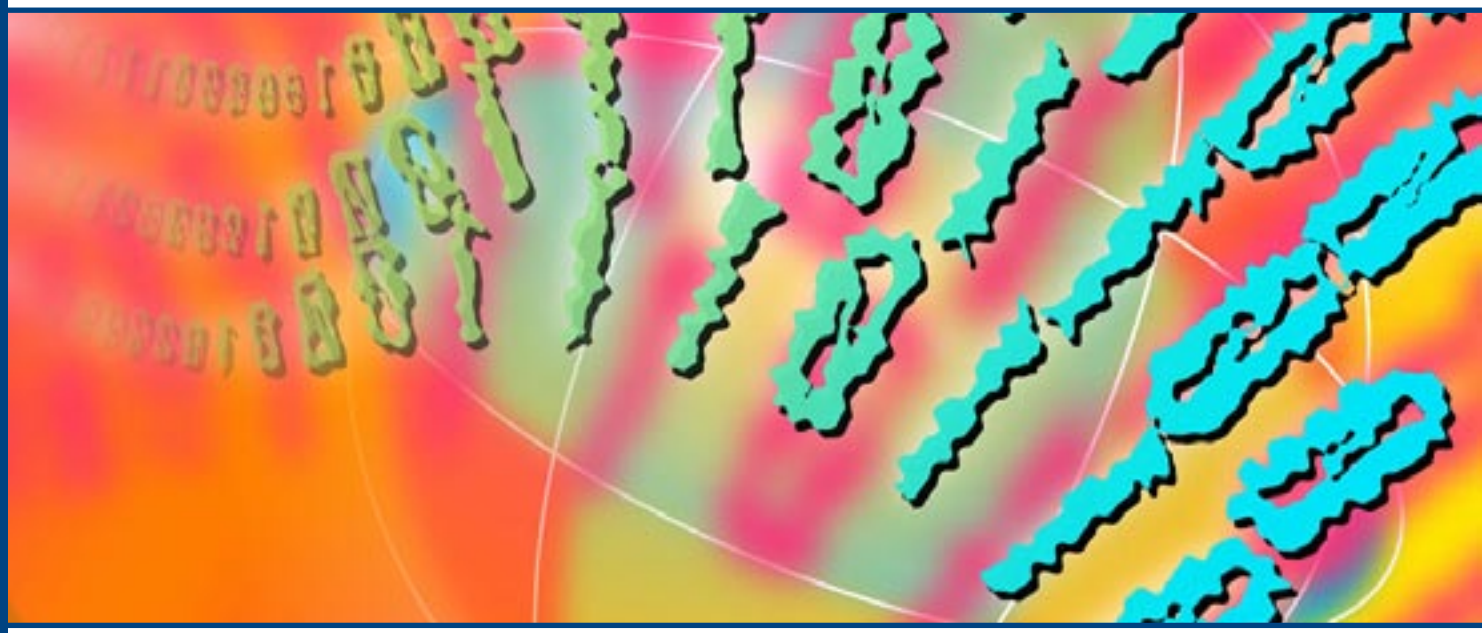
Natural from Software AG

Philip Howard & Tel Hudson

January 2006



an evaluation from

Bloor Research

...optimise your IT investments

Contents

Fast facts

Natural was one of the first 4GLs (fourth generation languages) to be introduced, over a quarter of a century ago. Of course, at that time it was mainframe-based, as was all serious computing. However, as other platforms have grown to take up a serious role within the enterprise so Natural has adapted to these new environments, so that it is now (and has been for some years) widely available on open systems.

In order to understand Natural's role within today's IT world, it is important to recognise the changes that have taken place generally with application development. In particular, not much new mainframe-based development is taking place and there is little demand for new 4GLs per se. On the other hand, there is a substantial demand for converting and upgrading existing applications so that they can run within a modernised environment, most notably in supporting a service-oriented architecture (SOA) and Web Services. This applies to both existing 4GL environments such as those provided by Natural and code-based environments such as COBOL, where it may make sense to migrate from code to a 4GL such as Natural in order to ease the transition to SOA. This is especially recommended if you want to be able to consume Web Services as well as provide them.

However, this is not to say that there is no new market for 4GLs. There are, in fact, three such. The first is new development for current Natural customers. The second is for legacy renewal of applications on the mainframe, while the third is a substantial market for 4GLs running on open platforms (including Natural) in emerging markets such as Eastern Europe, the Far East and South America, where Software AG has the advantage that it can offer a genuine alternative to IBM and Microsoft.

This report is in two sections, where the first addresses the functionality of Natural in general terms and its use for service-oriented application development, while the second specifically focuses on legacy renewal and application modernisation.

Key findings

In the opinion of Bloor Research the following represent the key facts of which prospective users should be aware:

- As a 4GL, Natural provides exactly the sort of reduced coding, simplified maintenance, and automated documentation that you would expect. It supports component-based development and supports a model-driven approach to application and service generation.

- A major feature is that the product includes automated error and transaction handling so that the developer does not have to worry about either of these issues.

- Natural also has strong features to support testing though its various tools have not yet been unified into a single product. This is planned for a future release. The product also includes resource monitoring capabilities.

- Historically, Natural was closely linked to Software AG's Adabas database. However, for years now, Natural has been available to work with your choice of database from DB2, Oracle and MySQL to IMS/DB and VSAM. As might be expected, the product also has strong support for XML environments.

- Configuration management, application management and software release management are all included within the Natural environment.

- For users wishing to migrate to Natural from legacy mainframe coding environments (such as COBOL and PL/1), Software AG has built relevant tools and set up a centre of excellence that can assist in this process. The company estimates that in excess of 90% of the conversion process is automated thereby.

- During the course of 2005 Software AG released a new version of Natural Engineer, which is a refactoring tool that can be used for code restructuring, code improvement, documentation, internationalisation projects and so on. The company believes (and we agree) that refactoring is a process that companies should go through en route to implementing a service-oriented architecture.

- A number of new facilities are scheduled for release in early 2006, which will expedite this migration to SOA. In particular, the company will be introducing Natural Business Services, which provides SOA development and management tools together with a Services Repository where all details of the services available to you can be stored and detailed.

- Also in 2006, Software AG will release an Eclipse plug-in and Visual Studio .NET add-in for Natural Business Services. In the second half of 2006, the company will introduce comprehensive Natural application development features under the Eclipse open-source framework.

- A graphical developer interface will be released in 2006 so that no direct web coding will be necessary to build rich internet applications.

The bottom line

There are various markets that Software AG addresses with Natural. The simplest is the traditional market for users that want a more productive, simpler to use environment for developing applications. It is our opinion that 4GLs still have significant advantages when compared to IDEs (integrated development environments) and we are disappointed that the market (at least in the West) does not seem to recognise this fact. Be that as it may, for those that appreciate this fact, Natural is one of the few such products that is both modern and available from a global vendor, which represents a significant strength.

For existing users of Natural, the decision to remain with the product, especially with the extensive SOA features that are being introduced, should be a no-brainer. Indeed, there is a good argument for extending the use of Natural beyond the mainframe if that has not been done already.

Finally, there is the market for code-based environments (COBOL, PL/1 and so forth) that need to move towards SOA. If this is simply a question of web-enabling existing screen-based applications, or even the provision of Web Services, then there is an argument for using special-purpose tools rather than Natural for this purpose. Indeed, Software AG's own ApplinX and EntireX products directly address this requirement. However, if you want to do more than just this; if you want to be able to build new service-oriented applications or add new functionality to existing applications then the advantages offered by developing in or migrating to Natural are significant and worth serious consideration.

Vendor information

Background information

Software AG was founded in 1969 in Germany. It originally made its name with the Adabas database, and subsequently with the Natural application development and deployment environment. From this basis the company has long focused on mission-critical transaction processing systems and this remains the case, even as the company has expanded its focus to service oriented architectures (SOA), service-oriented application development, modernisation, and process-driven integration.

For many years the company was led by its founder, Peter Schnell, and remained a technology-driven company that almost deliberately seemed to eschew marketing. However, he left the organisation in 1996 and the new management team has been more upbeat, floating the company on the Frankfurt stock exchange in 1999. As an indication of how late the company was in coming to market, it was (at that time) the largest IPO for a software company in history.

In the late '90s Software AG focused its attention on the XML market and this led to an emphasis (with hindsight one might say overemphasis) on its Tamino XML database. While this led to the company having its best ever year in 2001, the dot-bomb meant that the company had to retrench and it has now reorganised into two main operating divisions: one of which focuses on Adabas and Natural (Enterprise Transaction Systems), and the other of which is based around its XML (Tamino XML Server) and process integration technologies to create a single XML Business Integration suite.

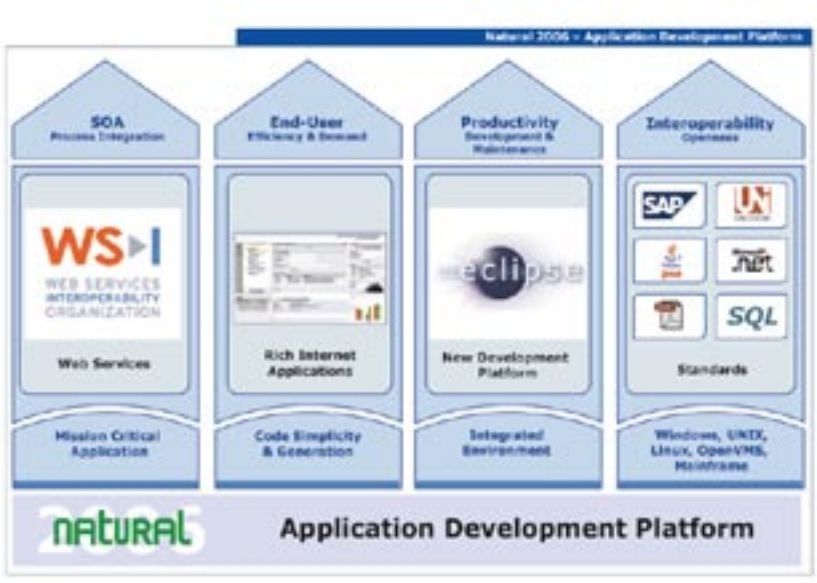


Figure 1: Natural Application Development Platform

Both divisions have a strong focus on supporting service oriented architectures (SOA) within the enterprise. Software AG is committed to openness and interoperability and the company continues to evolve both Adabas and Natural in

this direction, while still being available on almost all the platforms available in the market. In particular, Natural will soon support the automatic development of new web services with its forthcoming Natural Business Services offering, as shown in Figure 1 (which also illustrates the support for rich Internet applications and the Eclipse framework). In addition, the XML Business Integration Suite has added an SOA repository that is at the heart of the service-oriented integration platform. These SOA initiatives should offer the benefits of standards-based development and integration while leveraging existing IT investments and keeping pace with new compliance mandates.

Software AG web address: www.softwareag.com

Product availability

There are two different versions of Natural, one for mainframes and one for open systems; the primary difference being that part of the former is written in Assembler, though they are compatible at the source code level. They are Natural version 4.1 (mainframe) and Natural version 6.1 (open systems). However, in both cases you actually do development within a Windows environment and then deploy to your chosen platform. As well as the conventional version of Natural, it is worth noting that the Natural Productivity Package Personal Edition (which comes bundled with Adabas for Windows) is free, so that you can try out the Natural development environment.

Platforms supported include z/OS, z/VSE, z/OS.e, z/VM, z/Linux, BS2000/OSD, MSP, Linux (Suse and RedHat), HP/UX, Sun Solaris, AIX, SCO, OpenVMS and various flavours of Windows. On the mainframe there is also direct support for CICS, IMS, UTM, Com-plete and so forth.

Financial results

Software AG has some 2,700 employees world-wide, with subsidiaries and distributors in more than 90 countries. Company revenue in the last reported year, 2004, was €411.4 million, which was slightly down compared to 2003 in terms of figures but which represented a marginal increase if based on constant currency rates. More significantly, net income increased more than tenfold during the year, from €7.1 million in 2003 to €77.2 million in 2004. The most recent quarter (Q3 2005) demonstrates a continually improving trend with year to date revenue of €314.5m compared to €298.6m in the same period in the previous year. Operating income (EBIT) was €67.5m versus €60.4m for the same period in 2004 (net of extraordinary items).

The Enterprise Transaction Systems (ETS) business line (Adabas and Natural) also continues to show increased license revenue with growth of 9% in the first half of 2005.

General product description

Introduction

The essence of 4GL development systems like Natural is that they hide system and data complexity. This results in two major benefits. The first is that it allows the developer to get on with interpreting business rules without having to worry about all the ancillary tasks needed to run a program. This has obvious benefits when a new application is being written. The second is that it is much easier to maintain and update applications that have been developed using Natural, not only because documentation is generated, but also because the automation built into the product means the developer has much less to worry about. This has particular relevance to mainframe application renewal, as we will discuss in the second part of this report.

Architecture

The architecture of Natural is illustrated in Figure 2 and we will discuss the various features of the product under the headings shown here. However, there are several points to note, which are that the user interface separates the end user experience (terminals, the Web Portal and so forth) from the Application Platform; that interoperability with external applications is encapsulated through Web Services or platform specific wrappers (for example Java and .NET); and that a specific data interface isolates access to databases and XML files. This modular design allows Software AG to implement new languages or database protocols without affecting the core Natural services.

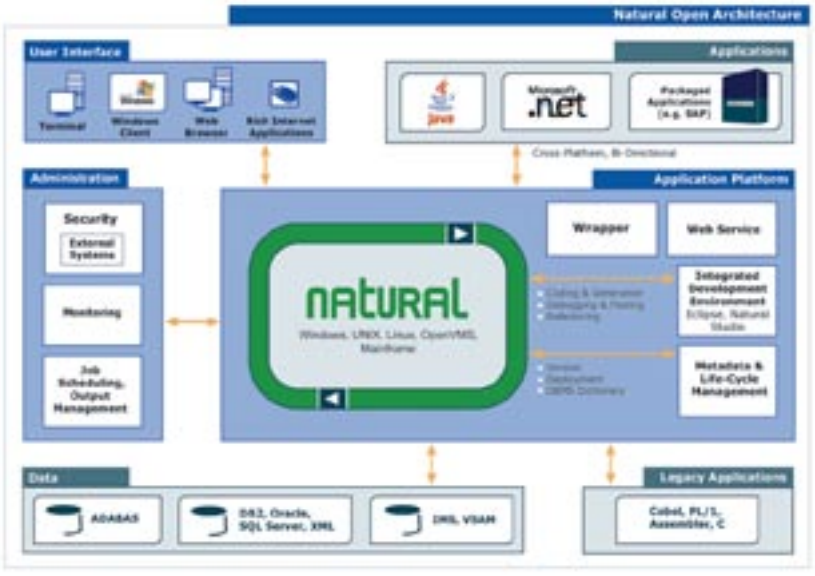


Figure 2: Natural architecture

In addition, as far as the user interface is concerned, this can either be application (form) driven or event-driven as is appropriate. As far as wrappers are concerned, there is bi-directional support for both Java/J2EE and .NET environments,

whereby there are built-in facilities to map Natural to the relevant wrapper and then that wrapper can be deployed. Wrappers are used to encapsulate Natural application logic and expose them as .NET or Java/J2EE classes. With the bi-directional capabilities of this wrapper technology, Natural is also able to consume .NET or Java/J2EE classes. Web Services are supported in the same manner and use the open integration standards SOAP, WSDL, HTTP and UDDI.

Language

The principal behind Natural's language is that programming is hard enough without the programmers having to worry about the complexities of the implementation as well as program logic. To this end as much of the implementation as possible is hidden from the developer. This has two additional beneficial effects. First, the programs are shorter and so there is less space to make a mistake. Further, when an error is found, there is less code to search. Second, because the implementation is encapsulated in the various interfaces, Natural applications are upwardly compatible and portable.

Let us take an example. Suppose that you want to access a database and then present the results graphically. In Natural you might create the following code:

```
HISTOGRAM EMPLOYEES LAST-NAME FROM 'JONES' THRU 'SMITH'
```

Now compare the equivalent Java:

```
empl.COUNTER.assign(0);

ResultSet RS$10076 = executeQuery("employees","select
  LAST-NAME,count(distinct ISN) from EMPLOYEES where LAST-
  NAME>='JONES' and LAST-NAME<='SMITH' and LAST-NAME is not
  null group by LAST-NAME order by LAST-NAME");

histogram_10077:

   try

   {

      while(RS$10076.next())

     {

        empl.fullName.lastName.assign(RS$10076,1);

        empl.NUMBER.assign(RS$10076,2);

        empl.COUNTER.add(1);
```

Clearly, it is a great deal more productive to use Natural. Further, both the catch statement and the end of the while loop have been left off the Java code.

Moreover, the Java code example also requires knowledge of SQL and if you change to a non-SQL database, the code will have to be changed. The Natural code simply requests information: it will work with any database.

Further, there is the question of the omitted catch statement in the Java code. Error handling is never simple and this says nothing about any particular language: it just happens to be a complex subject. In this example, the Java code is held within a try block. In the event of an error, the code will jump to the appropriate catch block. Having processed the error, the catch block returns control to the main code. For safety, the developer must remember to put every database access and any other statement that can fail in a try block. Each try has to be associated with a suitable catch. Because of scoping rules, most try blocks have their own catch (COBOL programmers should read this as 'On Error Goto catch'.)

In this Java example, the problem is that it is easy to forget to put in a try. It is even easier to forget the catch. This leaves the code unprotected. If the try is missing and an error occurs, you get an operating system message and the program may crash. If you omit the catch then you may get an unhelpful operating system message and the program will almost certainly crash.

Natural protects all database accesses, automatically handling all exceptions (and transaction handling for that matter). There is a common facility that catches all errors by default. If necessary, the developer can overrule the default and put in as many or as few error handlers as he chooses. Ultimately it is possible to have an individual error handler for each access. This is safe because in case of an error, an error handler within the application will be called. Even if this cannot recover from the fault, at least it should provide sensible diagnostics.

Development Studio

The Natural Productivity Package provides an integrated development environment for Natural. It is at the heart of the Natural system and what you see is a single set of tools running under Windows, regardless of the environment you wish to deploy to. This means that you do not have to learn several flavours of tools.

The Natural Productivity Package, which is a combination of Natural Studio (Natural for Windows and Natural Development Server), Adabas for Windows, and Natural Engineer, was launched by Software AG last year to support the demand for a better return on assets (ROA) when using Natural applications. Software AG's Natural Productivity Package is intended to help organisations to improve developer productivity to more quickly meet business requirements. With this Windows-based development environment, developers should get all the benefits associated with the use of modern desktop tools (improved productivity, increased quality, minimised risk, greater flexibility and so on, all of which contribute to improved ROA) along with support for undocumented and unstructured code within the production environment and while enabling non-mainframe programmers to develop and maintain mainframe applications.

The Natural Productivity Package features Application Development and Application Maintenance Tools. With the Application Development toolset, formerly Natural Studio, developers can perform remote development and maintenance of applications on any platform—UNIX, Linux and even the mainframe—directly

from the PC desktop. The Application Maintenance Tools, provided by Natural Engineer, provide versatile code analysis and restructuring tools which enables developers to quickly and securely maintain, re-engineer, and refactor (see later) very large and long-lived business applications.

Double byte code is supported throughout but not Unicode. However, the company plans to introduce support for Unicode in its next release for both mainframe and open systems, which is due during the first quarter of 2006. This will include tools to migrate from the existing environment to the new one. It is also worth noting that there is a text modeller utility which allows you to define your own language mappings so that a user interface in, say, French, can be automatically generated along with English versions.

A further function built into Natural is that code is checked and partially compiled whilst it is being entered. Consequently, you can test the application without having to wait for a build cycle. The result is that it is possible to immediately deploy a development version of an application for testing. This can be done without having to go through a compile cycle.

As far as editing is concerned, there are different editors for different object types. Thus there is a code editor, for example. Here, the editor distinguishes grammatical features by assigning different colours to different types of tokens. As an example, the standard options draw data names in black whereas language key words are blue. There are all the sorts of functions you would expect, including automatic formatting.

As stated earlier, code can be tested by a developer at any time. Whilst this simplifies the development process, it is not good enough for formal testing. Formal testing requires the code to be fully exercised. Natural does not support automated testing but it does have a number of tools that help, including the debugger (which supports local and remote operation, breakpoints, watch points, the ability to dynamically change variables and some code coverage capabilities), profiling capabilities and the built-in scheduler, which allows you to run a development application using a set of test data several times a day so that errors do not creep in. As far as testing tools per se are concerned, these are not currently integrated in a single automated testing environment, but this is on the company's roadmap and it intends to introduce such capabilities during the course of 2006. Software AG has a partnership with a third party for providing Web Services testing.

Eclipse

Software AG will expand the choices available to its developers by providing the Natural Productivity Pack (development environment) in Eclipse (this is targeted at the second half of 2006). Eclipse is a strategic development platform for Software AG. Plug-ins are already available for ApplinX and EntireX. Other Software AG development tools, such as Natural Business Services, will also be available in Eclipse and will provide a common user interface across Software AG's product portfolio as shown in Figure 3.

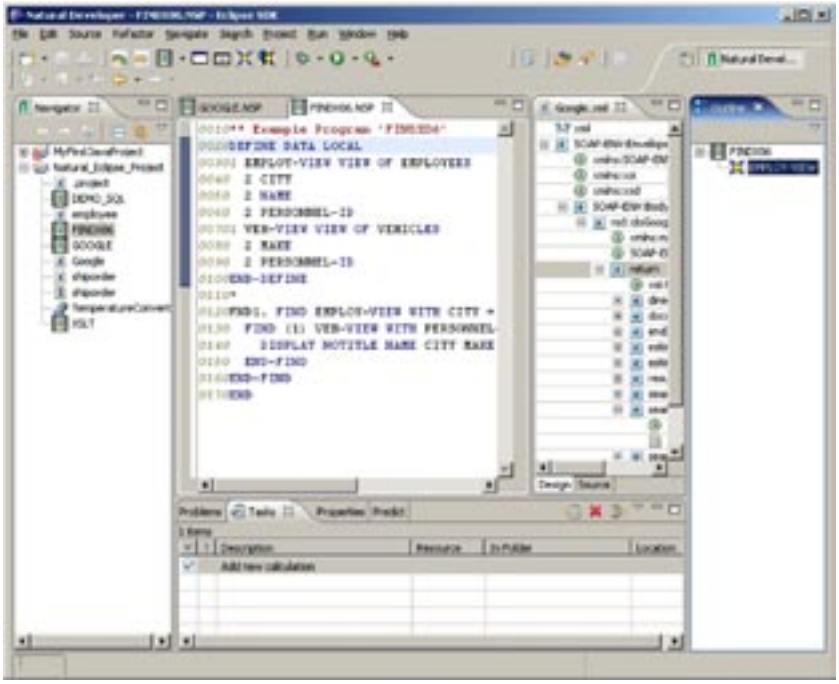


Figure 3: Natural Developer in Eclipse

According to the Eclipse organisation (www.eclipse.org) "*Eclipse is an open source community whose projects are focused on providing an extensible development platform and application frameworks for building software. Eclipse provides extensible tools and frameworks that span the software development lifecycle, including support for modeling, language development environments for Java, C/C++ and others, testing and performance, business intelligence, rich client applications and embedded development. A large, vibrant ecosystem of major technology vendors, innovative start-ups, universities and research institutions and individuals extend, complement and support the Eclipse Platform.*"

Software AG has decided to provide an Eclipse plug-in for the Natural Productivity Pack and many of its other products due to the rapid adoption of the Eclipse framework and the benefits it provides to users. With a common look and feel across all products, skills become more universal. Because functionality comes in more granular components, users can add specific units (plug-ins) that correspond to their needs. With open interfaces and common metadata, plug-ins can be embedded by partners and customers. A key benefit for Software AG is the additional exposure given to Natural as Eclipse is of growing appeal with the development community. Indeed, it has established itself as the de facto standard for non-Windows development environments.

Service-oriented application development

To support the need for service-oriented application development, Software AG has developed a set of services and facilities that are available as a part of the Natural environment. This new functionality will provide greater automation, enabling users to define web services based on existing sub-programs and generate brand new web services. This new set of capabilities, built upon the development studio

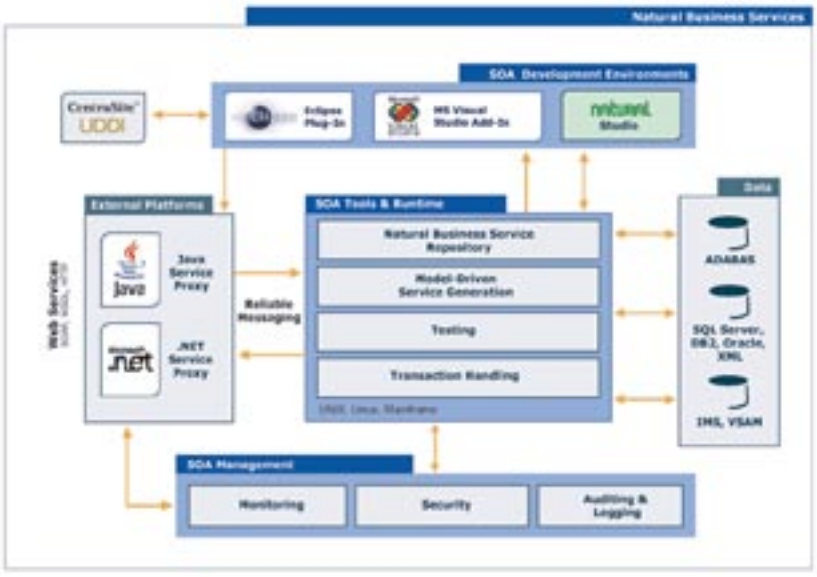


Figure 4: Natural Business Services

and model-driven code generator is referred to as 'Natural Business Services'; these are illustrated in Figure 4.

There are a number of points to be made about this offering. The first is that this represents the direction in which Software AG is heading as much as it represents current capability. In particular, the Eclipse (as discussed in the previous section) and Visual Studio plug-ins, as illustrated in Figure 4, and the Service Repository, will all be introduced during the first quarter of 2006. The second point to note is that the model-driven code generation of new services and the transactional control of service aggregations (so called service bundles) combine with both a service security model and end-to-end monitoring to allow the construction of mission-critical business services.

In so far as the plug-ins are concerned, these will provide functionality for Natural Business Services in Microsoft Visual Studio .NET and Eclipse (Java) development environments. This will allow the retrieval of business services from the repository and the ability to generate service-proxies and web services from the development environment of choice.

New web services can be created using Natural Business Services, which incorporates Natural Construct, a model-driven code generation platform. It provides an environment that generates applications and web services automatically, based on pre-defined models (that you can customise) for database access, business rules, user interface navigation and so on.

The Services Repository will be a key part of Natural Business Services when it is released early in 2006. The point about the Services Repository is that once you have ported your code to the Natural environment (if necessary) and refactored it, the actual process of creating web services is relatively straightforward. However, understanding the services that you have created, and how they can be deployed and reused is another matter entirely, which is where the Services Repository comes in. As can be seen in Figure 5, the Services Repository describes each

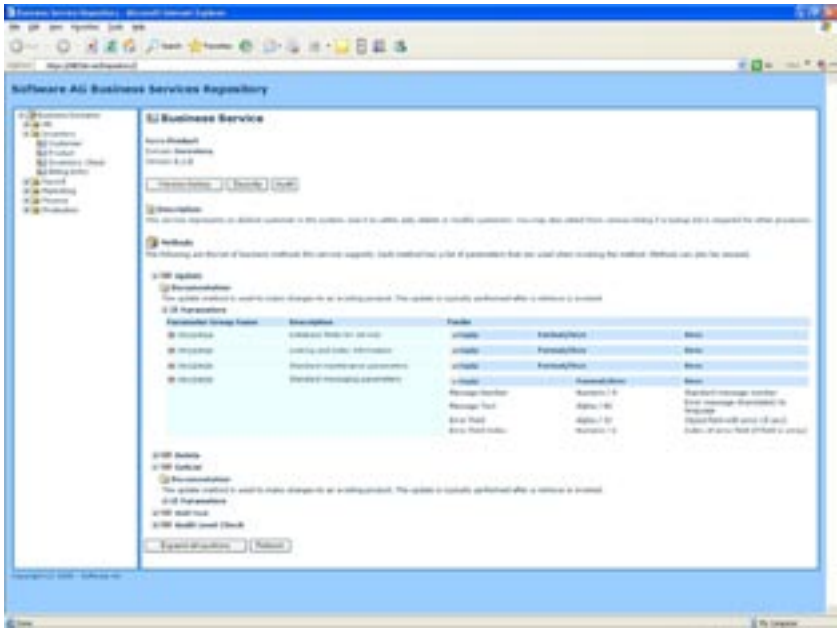


Figure 5: The Services Repository

service, the methods that it supports, and the parameters and fields to which it relates.

The Natural Business Services Repository stores and manages all metadata around services. It contains metadata about design-time (for example, service interface parameters and descriptions), as well as run-time information that is relevant for auditing and billing purposes.

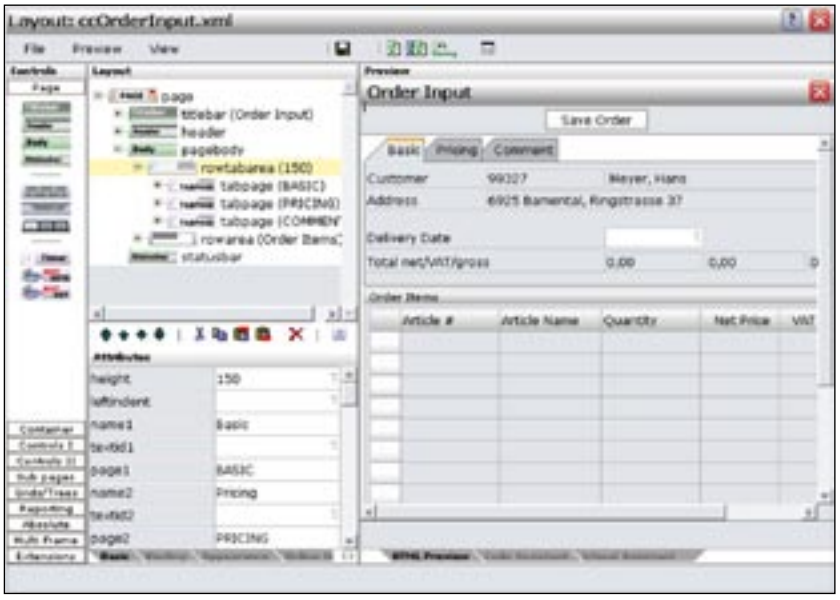


Figure 6: Composite Application Integrator

Software AG offers CentraSite, an open SOA repository for service-oriented environments. This is intended to promote greater collaboration between business and IT by uniting metadata from various service- and process-oriented integration products, such as Software AG's Enterprise Process Manager, Enterprise Service Integrator, and Enterprise Legacy Integrator. CentraSite

provides standard access (UDDI, WebDAV, XQuery and so on) for Web Services and is integrated with the Natural Business Services repository. What this means is that services developed with Natural can be published in an open SOA repository supporting UDDI.

Rich internet applications

User interfaces also have their own editors and Software AG provides tools to design the various types of user interfaces. These include terminal application and windows dialogs as well as web pages (what Software AG refers to as 'rich internet applications').

Designing web pages is an artistic occupation that attracts a different personality than the programmer who can wade through complex business rules. The web editor for Natural, Composite Application Integrator, shown in Figure 6, can be used to connect the fields in a web page to items known by the program. In this way, the programmer can tell the artist what fields are required and get on with coding. When the web page is available, it will automatically work with the program. Similarly, if cosmetic changes are made to the user interface, there is no need to touch the code. The new page is simply substituted.

For so-called 'rich internet applications' Natural includes a graphical developer interface so that no direct web coding will be necessary. This should obviate the need for web experts. The big advantage of this sort of approach is that the resulting application will interact directly with Natural so that, for example, you can make use of the power of the Natural debugging system to solve problems related to rich internet applications.

Configuration and application management

Predict Application Control and Predict are the parts of the Natural environment that provide application, metadata and configuration management, supporting team development and both off-line and on-line development.

When working offline, developers copy the entire project onto their own system. From time to time, each developer must synchronise his version with the main project and this is done automatically when the developer logs into the central system. The advantage of this method is that a developer can work on a remote terminal that is not connected to the main computer.

There is only one copy of the program when developers work on line. A developer gains ownership of an item and edits it. When he is happy, he makes it available for the rest of the team. This means that no synchronisation is required but it does mean that the development station must be connected to the main computer. In both cases (on-line or off-line), the editing is actually done locally on the developer's own machine.

From a configuration management perspective, Predict Application Control provides the sort of facilities one would expect: check-in and check-out, versioning (including the automatic creation of new versions), support for baselines (used by the software's release management) and so on. There is also a cross-referencing facility that acts in a where-used/impact analysis capacity.

Another benefit of Predict Application Control is that by controlling the systems development lifecycle and ensuring that program changes are tested, validated and authorised before moving into production, IT Departments can more easily comply with government regulations such as Sarbanes-Oxley. The controls enforced by Predict Application Control support good IT Governance practices advocated by leading standards organisations such as COBIT and ITIL. Only one change management system is needed with Predict Application Control to control the migration from testing to production of Natural, COBOL, JCL, and Assembler objects.

Interoperability

Natural applications are portable. If you write an application that runs on one platform then it will run on all the other platforms that are supported. The problem then is to link your application to libraries written in another language.

Natural and COBOL can each call sub-routines directly from one another or may be linked through wrappers. Other foreign applications such as Java and .NET are linked through wrappers. A new wrapper can be written and added to the system without changing the Natural core. Consequently, the range of supported foreign programs will continue to increase. The wrappers support bi-directional transactions. That is to say, Natural can act as either a client or a server. One of the important supported groups is the .NET family. Because .NET is a unified system, if you have, say, a J++ wrapper, then you automatically have wrappers for C++, Visual Basic and the other languages.

Developers can also work in the environment with which they are most familiar. Developers can access Natural from within the .NET IDE; Natural adds an extra menu item to Visual Studio. Java and J2EE are linked but in this case, it is through Eclipse.

As far as databases are concerned, Natural was historically linked with Software AG's Adabas. However, it is important to appreciate that these products can also operate independently so that the company has implementations of Natural in conjunction with Oracle and DB2, for example. There is a special version of Natural for DB2 on mainframes, which has been optimised for access to DB2. More broadly, access is supported to all the leading open source and mainframe databases, including Sybase, SQL Server, Informix, VSAM, IMS/DB, MySQL and so on, as well as DB2 and Oracle. As one might expect from a vendor calling itself "The XML Company" there are also significant capabilities for parsing and manipulating XML from within the Natural environment and, of course, you can access the Company's Tamino database directly. Again, it is also not surprising that there is an especially close relationship with SAP, with the ability to transparently

call SAP R/3 BAPI, RFC, IDoc and ABAP code or exchange messages with the SAP NetWeaver Exchange Infrastructure (XI).

Data is handled by a simple system. Data items are defined through the DDM (Data Definition Module) and then transactions are defined using DML (Data Manipulation Language). Both the DDM and DML are independent of any type of database. In addition, users may directly use SQL statements. Accordingly, Natural treats all supported database types without the developer needing to know anything about them.

However, look at the following code:

```
TAMINO. READ VW-TAMINO BY NAME
  SQL. FIND(1) VW-SQL WITH PERSONEL_ID = PERSONEL-ID (TAMINO.)
    NAME := NAME (TAMINO.)
    CITY := CITY (TAMINO.)
    UPDATE
  END-FIND
END-READ
```

This code runs through a Tamino database. For each entry, it finds the entry in a SQL database with the same name as the Tamino entry. It then updates the SQL entry using Tamino data. The importance of this code fragment is that it shows two different databases being accessed together with data from one being used as a key by the other.

Administration and Security

There are a number of elements to administration, including both system management and security.

Entire System Management is a portfolio of tools available to monitor, schedule and secure the data centre environment. The product family consists of a modern user interface that is used to control data centre resources and services, automate job scheduling and event-driven batch-processing (24x7), automate output organisation, and optimise print-processing (24x7). In addition, a console that monitors events within the data centre and an interface to all system information and services, are both provided.

In the case of security, Natural can add an additional layer of security on top of that of the operating system. Any user, user group or user type can be given a set of permissions, which allow or restrict access to the Natural system. In addition to restricting access to data and applications, Natural can be configured to restrict usage of libraries and utilities. Note that it is this approach to security that allows Natural to permit remote debugging. If a developer wishes to check a live application that is running then he must have enough privilege to access the

debug libraries and utilities. We should assume that the project manager would be very careful about assigning these rights.

Natural also supports a number of external security systems including RACF, TopSecret, and ACF2. LDAP support is planned for 2006.

Renewing existing environments and applications

There are a variety of different ways that existing applications may need to be upgraded to take advantage of modern capabilities such as the Internet and service-oriented architectures. For Natural users this will typically mean using the features that Software AG is adding, or has added, to Natural, such as the support for rich internet applications and refactoring. However, for both Natural and non-Natural users there are three options offered by Software AG. If it is simply a question of SOA-enabling (as a provider) or web-enabling your existing environment, then the company offers a variety of modernisation approaches from non-intrusive to bi-directional. If you are interested in transforming an existing application to Natural for new development, Software AG offers conversion services. Third, re-factoring is available for optimising applications to set sub-programs up for re-use with Natural Business Services.

Legacy web services and web-enablement

Software AG's enterprise legacy modernisation portfolio includes ApplinX and EntireX. ApplinX provides a non-intrusive approach to SOA or web-enabling with no need to understand or change the application code. EntireX offers bi-directional communication support, not only exposing legacy transactions, but also allowing legacy programs to consume .NET, J2EE or Web services.

Legacy transformation

However, if you want to be able to develop new functionality (for example, to be able to consume web services on the mainframe as well as merely providing them) then it may be worth considering a migration to Natural. If that is the case, then Software AG has set up a conversion centre, in the UK, and developed a number of conversion tools that will convert your applications from PL/1, COBOL or Computer Associates' Ideal to the Natural environment. The company claims that between 90% and 95% of the conversion process is automated through these tools and, if true, this is an extremely good percentage. Moreover, the software should actually end up as higher quality since Software AG's tools have the ability to identify redundant sub-routines and then eliminate them automatically (this is part of the refactoring process discussed in the next section). Of course you also get documentation, reuse, easier maintenance and all the other advantages you would expect from a 4GL.

Refactoring

A new release of Natural Engineer specifically designed for refactoring provides the facilities illustrated in Figure 7. Refactoring with Natural Engineer is used to improve and optimise the structure of an application and, at the same time, make that application easier to maintain. So, for example, you might want to change the name of an object and have all references to that object automatically

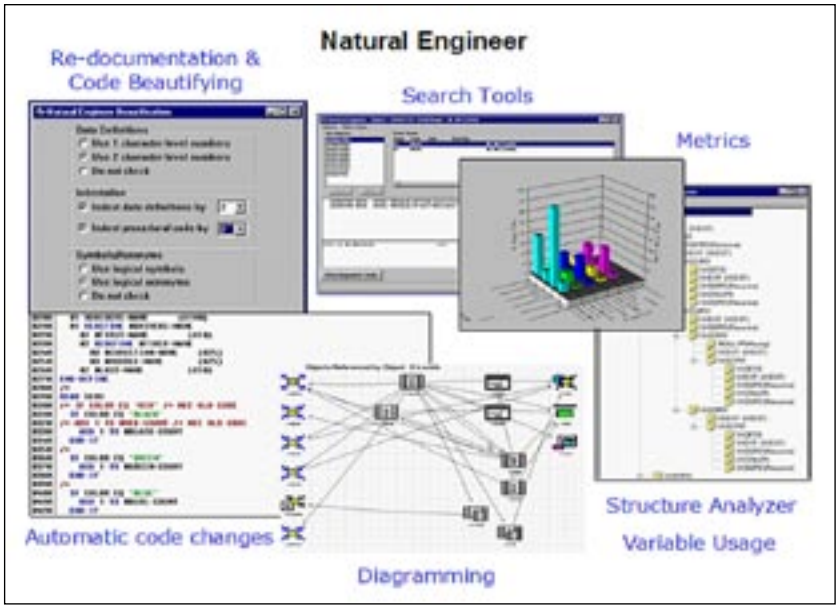


Figure 7: Facilities in Natural Engineer

updated, or remove unused fields and redundancies, or identify similar sections of source code that offer potential for simplification, or optimise statement collections, or whatever. Software AG's view is that this is a necessary pre-cursor to coding service-oriented applications. While, from a theoretical perspective, it is not actually necessary to do this we are inclined to agree that, in practice, this makes good sense.

Code analysis, the first step to refactoring and improving code quality, is also provided by Natural Engineer. It allows the project administrator to set coding standards for this project (though it is always possible to construct coding standards that cannot be checked automatically) and thereafter Natural requires that code should conform to these standards. Natural can also check items such as the format of variable names and it is also possible to restrict the use of some program constructs (a classic case of this is the traditional proscription of goto). Natural Engineer also provides code analysis and re-documentation for COBOL and JCL.

Product futures

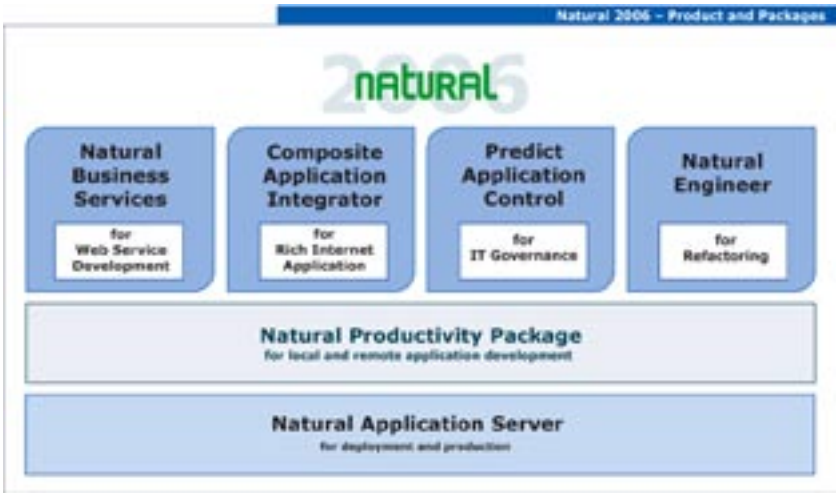


Figure 8: Planned features for Natural

Software AG has announced both short-term and longer term plans for Natural with key areas of focus illustrated in Figure 8. The former have generally been discussed already as they apply to the first quarter of 2006. In terms of longer term plans, the initial focus of the company is on:

- Rich internet applications.
- SOA, Web Services and XML.
- Eclipse support across the product.

Further development will focus on:

- Additional refactoring capabilities.
- .pdf support for reporting and documentation.
- A messaging gateway.
- Extended J2EE/Java interoperability and Microsoft .NET interoperability.
- The integrated testing environment alluded to earlier.
- The introduction of visual modelling and support for model-driven development (though this is unlikely to be MDA-compliant).
- Application portfolio management.

Summary

The listing of areas in which Software AG intends to enhance Natural over the next few years is, we believe, an important indicator that demonstrates the company's commitment to this product. More than that, it shows a determination to ensure that Natural remains suitable for use within a modern IT environment, serving the needs of modern IT developers. This particularly applies to service-oriented architectures where the company is implementing extensive capabilities on all of its supported platforms. SOA is clearly the direction in which the market is moving and the combination of support for Web Services, refactoring, rich Internet applications, IT governance and other capabilities that are being delivered by Software AG should be a winning one.

We are impressed by Software AG's approach. All too many (though by no means all) of its competitors are downgrading their efforts to maintain their 4GL products, regarding them as non-strategic. In our view this is a mistake. We believe that Natural has as much relevance as it ever did: it does not require detailed knowledge of complex environments such as Java and J2EE, it is more productive (for both new developments and maintenance), better enables reuse, and so on. It is pleasing to see that Software AG clearly takes the same view. As long as this remains the case then the future for Natural users should be assured.

Bloor Research Overview

Bloor Research has spent the last decade developing what is recognised as Europe's leading independent IT research organisation. With its core research activities underpinning a range of services, from research and consulting to events and publishing, Bloor Research is committed to turning knowledge into client value across all of its products and engagements. Our objectives are:

- Save clients' time by providing comparison and analysis that is clear and succinct.
- Update clients' expertise, enabling them to have a clear understanding of IT issues and facts and validate existing technology strategies.
- Bring an independent perspective, minimising the inherent risks of product selection and decision-making.
- Communicate our visionary perspective of the future of IT.

Founded in 1989, Bloor Research is one of the world's leading IT research, analysis and consultancy organisations—distributing research and analysis to IT user and vendor organisations throughout the world via online subscriptions, tailored research services and consultancy projects.

Copyright & Disclaimer

Suite 4, Town Hall, 86 Watling Street East
TOWCESTER, Northamptonshire, NN12 6BS, United Kingdom

Tel: +44 (0)870 345 9911 – Fax: +44 (0)870 345 9922
Web: www.bloor-research.com – email: info@bloor-research.com

…optimise your IT investments