What is Diaxon DevX?

DevX represents a major evolutionary step in business application design. What does this mean?

How does Diaxon DevX deliver real benefits to you over and above those of other suppliers?

1. We Understand Your Requirements

DevX ensures that the application's logical structure and infrastructure is created correctly before the business-specific processes are built.

We perform a Strategic Business Analysis (or SBA) – a crucial first stage to ensure the application we will build for you is going to deliver the most benefit for least amount of development time and money – in common parlance, "the most bang for your bucks". We include a cost benefit and analysis in the SBA so you decide what the essential are and priority elements needed.

As part of this process we need to understand the two most important ingredients of an application: the functionality required and the data needed to support it. In essence you input some data, process it in some way and then re-present it in some other form. For example, e.g. a customer orders some of your stock so you enter customer and order details (products, prices, quantities), then process them to produce a sales invoice. The art of application design is knowing precisely what functions are required and what the exact form and content of the data should be to fulfil them.

Every application has a 'logical structure' that determines how the information used is 'packaged'. Data comprises a number of related components commonly referred to as 'entities' (examples are: customer, product and order). The logical structure which describes how entities are interrelated is commonly known as an Entity Relationship Model or ERM and creating an accurate ERM is an absolutely essential precursor to design – derived through careful data analysis including discussions between the application designer and the client. ERM structure and shape exactly supports the functions the application must perform and these are determined by the scope and specific business requirements – usually in a "functional hierarchy". Deriving both the correct ERM and functional hierarchy is fundamental successful application development and it forms the bedrock of our methodology.

2. Then we ~~create~~ generate your application

So what's different about Diaxon's DevX, doesn't every designer start like this? Well, the difference is that once we have QA'd the ERM and functional hierarchy to our exacting standards, we automate. Only when the functional hierarchy and supporting logical structure (ERM) is fully established do we proceed to the detailed design phase and the main substance of the development work. If the data structure which supports the functionality is flawed, wasting time (and money) later down the line is unavoidable as inappropriate or incorrect functionality would be built – so we aim to get the structure correct and in its most simple form. Unnecessary complexity is avoided at all cost. Care and attention at the beginning of the process pays dividends by avoiding costly corrections, amendments and delays later.

Automation!

Even quite diverse applications have much in common and many of the underlying functions and structures in any one application are very similar to that in any other. For example the 'user menu' and data maintenance screens have the same structure in most applications. DevX technology replaces a huge chunk of build development as it generates repetitive, predictable structures automatically without designer or programmer intervention. DevX creates all code for components that control the common infrastructure and architecture - automatically – representing a major saving in time (and money) which would otherwise entail hours and days of specialist development work. It also guarantees a consistently high and controlled build standard.

Although initially in a 'bare bones' form, the completely auto-generated application nonetheless comprises all the key structural components that form the 'chassis' of your application, comprising:

- The supporting database
- Systems architecture
- Menu structure
- Display, entry & maintenance screens
- Standard applications functions
- Any data interfaces
- Functional 'stubs'

Generated in our own house-style (the 'look and feel'), even at this stage the application, is *fully operational and has full data content and integrity*. It behaves like a full application and it can be viewed /tested by the client as required. If it is a web-based 'clouds' configured application you will be able to access and display both the system and its data directly through your web browser.

This early view of the application even though business-specific functionality has yet to be added, is very useful to both Diaxon and you the client. A lot of basic testing can be done and any flaws discovered in either the functional hierarchy or the ERM can usually be easily identified – a distinct advantage over traditional development methodologies in which flaws and omissions can persist unnoticed until very late in development. Furthermore, because DevX generates an application in a matter of seconds, we have the ability to delete the current version, make any corrections or amendments to the specification and regenerate the application. It only takes a few minutes to generate/regenerate the application and we can repeat this process on demand until both ourselves and you the client agree that the overall structure really is correct. As we will stress later – this is not traditional RAD technology, the application is not a rolling prototype, we are not out to develop an application for show or in front of our clients – the application is the real thing – or rather the chassis for the real thing.

3. Adding the specialised functionality

Once we have tested and made any amendments to the chassis comprising the outline functionality and data structures, we have reached another crucial QC point we can press ahead with further detailed design and build i.e. add 'flesh to the bones'. These specialised functions and processes – eg. transactional processing, user output (reports, invoices, interface I/O and other processes) may be unique to your application and require original design or they may be generic processes which we already hold in our 'knowledge base'. DevX has a library of reusable components so even the more specialised functions may already be available as reusable objects which can be used or adapted for your application.

Your application's ERM and functional hierarchy are the master specifications common to the whole design and will be shared between different development lines so separate business processes teams may be working, in parallel on different components. And, as modules become ready for testing they are plugged into the application's chassis. Production of these modules too are highly automated and use code generators to build functions that can be decoupled, amended, regenerated then reattached making the build phase highly modular.

4. 'Finishing' your Application

The last step in the build process is the 'finishing phase' when all interdependent functionality has been completed and tested and there is no more benefit in auto-generating components (automation can only be exploited so far and is used to create repetitive and predictable components). The purpose of the this last build phase is to add the unique styling finishes (e.g. graphics) and complete any functionality that really does require specialised programming or other development skills. If you don't want to use our house style – this is when the 'chroming' is done – putting on the bells and whistles and applying your own style and imagery.

More About Diaxon DevX Methodology

Logical and technical problems are tackled at the 'least costly point' or LCP.

Although a much overused expression, DevX really does employ 'holistic' design methods which fundamentally require a 'top-down' rather than bottom-up approach. Top-down ensures that all components are identified and are constructed in the correct order and are structurally correct (Quality assured) before the next stage can be started.

Using a common production-line metaphor, think of your application as a car being built along an assembly line. Once the basic chassis has been built it starts its journey at the beginning the line and manufacture is a highly engineered process of adding components in an orderly, sequential manner. Major basic structural components may be added by robots and later down the line, customer specific components will be added by specialised production-line workers. All components are added in their own time and place – carefully thought out by the assembly line designers. The same can be said for Diaxon DevX.

Our production line methods could not work without very high standards of quality assurance – using the above metaphor, if there was something misaligned on the chassis, at some point further down the line the fault would be spotted – and the later it was spotted the more disruptive and costly it is to fix. This phenomenon is well known to the IT industry! To avoid the assembly line coming to a halt and disrupting yours (and potentially other projects), we aim to get the design right first time. The I.T. industry, by and large, is still to learning these lessons and is a long way off the smooth and efficient production principles perfected by the Japanese car industry in the late 20th century.

Although superficially similar, DevX is not a RAD (Rapid Application Development) methodology in the accepted sense. We do not go in for prototyping nor do we strive to arrive at an accepted design through intentional iteration. Although often giving the illusion of early progress (you have something to show the client very early on), the traditional RAD often leads to a costly re-engineering process and a 'rolling prototype' with much friction developing between commissioners and developers as that apparent initial rate of progress falls away rapidly.

Diaxon and its proprietary DevX technology re-addresses this problem and puts the 'get-it-right-first-time' objective or GIRFT ideology' at the heart of its product development and delivery process to minimise iteration prevalent in the RAD approach. Because so much of our build is automated, our design input requires exacting standards of quality, consistency and integrity so we devote a much higher relative proportion of our time up-front in analysis and design.

Sure footedness and the GRIFT ideology deliver the key benefits of quality assurance, timeliness and therefore minimised costs – all of which are passed to you the customer.

Segregating Development Activity

Returning to our car production line metaphor, as the chassis passes down the main assembly line, there is as much activity away from the line as there is on it. Component assemblies are being constructed elsewhere, often by Third Party companies away from the factory. These assemblies have to be completed and delivered JIT (just in time) at a predetermined point on the line. And so it is with DevX, functional components, that share the same master ERM, can be assembled independently by other specialist staff (or even external to Diaxon).

Most contemporary software development companies require people with extraordinarily broad skill sets, gifted and highly experienced people; however, these abilities are naturally costly and sought after. DevX deliberately moves away from the traditional 'craft' approach to that of the production line, our aim is 'compartmentalise' skills – so rather than multi-tasking craftsman being able to build whole or much of the item, and we train and employ people with particular and specific duties. Not only can we reduce costs but re-skilling the development has the effect of driving up standards ensuring each job is done to the correct quality level – we replace individual idiosyncrasies with consistent and standardised code. For example one of the most important tasks in development is the 'Relational Database Design' which quite often is performed in a perfunctory way. Diaxon is a specialist in database design and by using highly trained specialists we ensure quality is delivered to our own exacting standards. Similarly, skilled graphics designers are used only when their skills and experience is needed. Likewise skilled 'logical' developers are not asked to do graphic design. Throughout the process the skills appropriate to the job are applied whether it be strategic business analysis, detailed data analysis or application designers. Because we are highly automated we no longer have need for basic grade programming – all segregated functions require specialist and specific roles which are under our full quality control.

Separation of your business logic (= your IP) from your application's infrastructure

A common problem with contemporary systems design is that business logic is usually dispersed around your IT systems. This is partly because, over time, enhancements and 'fixes' are inserted here and there ….that's how business evolution works BUT also because systems are constructed from components sourced from different providers. It's tempting to incorporate some of your 'business rules' in database 'constraints' – e.g. input validation rules. It's also common to put business logic into screen/form technologies and of course there's logic in algorithms and transactional programs.

DevX takes a different approach we intentionally keep all business logic separate from application infrastructure. Helix is a fully integrated high level, business-oriented language. Helix allows functions and modules to be developed and once generated, quickly slotted into the application at the appropriate time. Component modules are developed independently and then 'assembled' to create the final product much like in the aircraft industry.

The Helix language runs through the core of DevX and is used to define / create business transactional processes. Diaxon DevX advanced approach to business processes (where much of a company's IP resides) ensures that all business logic is held in just one form – it is not scattered through other application components (e.g. in screens, database constraint & 'trigger' logic).

Summary:

Diaxon DevX's production-line approach to development delivers real benefits over and above other contemporary methods by:

1. Vastly speeding up application delivery.

Standard structural code is automatically generated from a logical schema (most applications have standard components, Menus, Functions, Entities, Relationships etc.). DevX automatically creates this code quickly and effortlessly.

2. Reducing costs by getting each stage right 'first-time'.

All too often in traditional development a process of iteration takes place until the final application is eventually found to the satisfaction (or not) of each party this is random and dangerous depending on the type of application. This can be characterised as a 'bottom-up' development method. In all but the most trivial of applications it just gives the 'illusion' of progress but in reality builds-up problems (side-lines them) which will have to be solved at some point, usually business relation logical questions.

3. Ensuring Quality, Consistency and Standardisation is maintained throughout the development process.

Our definition of **Quality**: "That an application is 'fit-for-purpose', that it does what it supposed to do and it does it with consistency.
Consistency: is "the code is delivered the same way over multiple forms, middleware and database design". **Standardisation**: "the look and

feel of the application is the same for all users across multiple applications and platforms, the user does not have to learn new ways of doing things as they browse from one application to another.

www.diaxon.com