

# What are the pros and cons of Uiface with regards to development tools like .NET and Java?

A Java developer (Jay), a DotNet programmer (Dee) and a Uniface consultant (You) walk into a bar (i think there's a joke in here ;-)). You order drinks and start talking about the qualities of the software that you use.

Right off the bat Jay starts arguing that Java is free and that both DotNet and Uniface require a paid license in order to use it. Dee agrees that although DotNet requires a paid license it has a good price to quality ratio. It also features a extensive list of free light versions, and 30 day trials that can be downloaded from the Microsoft website.

Both Jay and Dee agree that your software has a license structure that is totally out of touch with today's demands. It is too expensive for small businesses and therefore is only being used in a small segment of mainly large companies.

You bring a good point when you tell them that of the three tools only Uniface is a 4GL, that has both platform and database independency that both of the other tools lack. You rub their noses in the fact that they are working with an inferior product.

The three of you almost try to decide the argument with your fists when a beautiful woman walks over and starts talking to you. (hey it is my fantasy, get your own!). She tells you that she overheard your conversation and that she is a management consultant who needs your advice. (still my fantasy...).

The job she is currently doing is writing a business case for a company where they are looking to replace there current information system with a new one. The choice is between Java, DotNet and Uniface. Part of the business case is a SWOT anlysis of all three tools. She asks you if you can help her by providing the arguments for either Java, DotNet or Uniface.

**Richard Walford**

Vice-President at Astaro AG

**With Java you will discover in three years time that you are still developing the application and if you ever want it to go into production you'll need to take a different route.**

**With .Net you will discover in three years time that you are on an unsupported framework and Microsoft are telling you to rewrite it if you want to stay in production.**

**With Uniface you will discover in three years time that the application is still working fine after all this time.**

**Arjen van Vliet**

Senior Consultant at Compuware

**From a 'Product Evaluation' paper by Bloor Research regarding Uniface 9:**

**"Arguably the biggest difficulty that Compuware faces with respect to Uniface is that development environments with a 4GL/RAD ancestry have fallen out of fashion.**

**We believe that this is a mistake. The sort of productivity gains to be attained from a product such as Uniface, both in initial development and on-going maintenance, are significantly more substantial than those deriving from code-based development environments such as Visual Studio .NET, Eclipse and other IDEs.**

**Moreover, these products do not offer the same guarantee of future-proofing (as proved by example in the case of Compuware) and, indeed, the history of some of the suppliers in this market suggests exactly the reverse.**

**In our opinion, far more attention should be paid to development products like Uniface, which deserve a resurgence of interest within the marketplace, not just in emerging markets but also more generally.**

**That said, Compuware continues to gain significant numbers of new customers for Uniface: not only does the product warrant such an investment, it merits even more widespread consideration."**

**Georges Moiny**

President of Orthosys

**The choice is simple : choose Java if multi-platform support is really important for the project, otherwise, choose .NET, which is the most**

**productive platform of the three. Uniface is not anymore a multi-platform application development tool, since the GUI is running only on Windows.**

**The real problem with Uniface is really the reliability of the runtime; even installing a minor release can disrupt an entire application. I would recommend that person to have also a look on a paper published by SPR (Software Productivity Research) in 2006, in which Uniface is compared to other programming languages : even .NET is considered more productive in that paper.**

**The feeling of being more productive with a 4GL is a mistake. You may have the feeling of being able to craft an entire application in a few weeks without design, but this approach will just lead you to unmaintainable software. So, you still have to design properly, whatever tool you use. You must also take into account that it is commonly accepted that the coding in a software engineering project account for only 30 to 40% of the time spend to develop the product. Therefore, the impact of the programming language productivity is limited. However, having an environment allowing easy refactoring and implementing common object oriented approaches will pay at long term.**

**Ron Hollander**

Software Engineer at Planconsult

**In Uniface it is possible to paint entities on your form. You can paint an outer entity and after that paint an inner entity within the outer entity which can be an up entity (many to one) or a down entity (one to many). The advantage of this concept is that it speaks to the imagination of beginning developers so they can get started more easily. The disadvantage of this concept is that relationships are now defined both in the component as in the application model (and most of the times also in the repository of the database being used).**

**If you are using Java in combination with Hibernate, relationships are only defined in the mapping files (and in the repository of the database being used). Navigating through relationships in your code can be done by code completion (aka auto completion), a feature Uniface doesn't have.**

**If it comes down to Uniface, I want to focus on the following things:**

- Association (outer entity is down entity, inner entity is up entity)**
- Composition (outer entity is up entity, inner entity is down entity)**

**It would be better and more flexible if Uniface stored all compositions and all associations an entity can have in the application model (current entity is outer entity). Currently Uniface stores all many entities an entity has in the application model (current entity is one entity).**

**When building a component it would be better if it works like this:**

- First choose all entities you want to use in that component from a list.**
  - After that choose the compositions you want to use in that component from a list (a list of all relevant compositions based on the entities chosen before).**
  - And finally choose the associations you want to use in that component from a list (a list of all relevant associations based on the entities chosen before).**
- The data from the lists mentioned before is derived from the application model.**

**Ivan Caspeta**

Software Architect at JDA Software Group

**Uniface provides a really nice head start when development starts since you can get up and running by just reverse engineering a data model or just modeling entities from the business case and you can see working screens take form in just matter of hours; but, as you move along in an iterative process or agile methodology, you are going to be throwing away a lot of that code because your architecture is settling down and you want to have your standards, templates and so on. On the other hand though, if Compuware provides a nice set of "general" use templates or the company has already invested or invests quite some time upfront to come up with rich templates and utilities/services, this could be minimized.**

**Interestingly enough, this is true for .NET and Java; developers can "hack" code away and get things up and running quite fast but when the architecture starts to settle and standards start to form, you will be either throwing away a lot of that code or re-factoring will be your friend, which touches an interesting point: refactor in Java/.NET could prove simpler as development tools such as Visual Studio and, say, IDEA/J for Java have quite some nice features for this type of task whilst in Uniface this might not be that much easier since the last time I used the IDE it didn't have such capabilities (Uniface 8).**

**I have not used Uniface 9, but a lot of the bells and whistles of .NET, for example, could be proven hard to mimic in a pure Uniface world and thus you will eventually be writing some C/C++ or C# to get to that level.**

**Now, Java is pretty much mature and there are tons of information/blogs/forums around it; plus the open source community has enormous amounts of nice tools. .NET also has a lot of info available and many things are in the open source world too. Uniface on the other hand is limited to the development houses that provide that type of service or Compuware and thus reducing the knowledge pool quite considerably.**

## **Dino Seelig**

IT Architect at ITIS Informatisering and Owner, Visual RoSe

**.NET and Java could be a good deployment platform when it follows the Model Driven- (MDA) and Template Driven Architecture (TDA) paradigm. With Uniface you can implement MDA and TDA. The hole-in-one building blocks shows that Uniface has a strong MDA/TDA base (sinds Uniface 7206). We follow Dr. Deming Plan Do Check Act cycle using Barry Boehm Spiral Software Process Model. With other words we have a strong focus on quality. In mine opinion, With Uniface you can build future proof strong stable business critical enterprise solutions. Handcrafting a Uniface, .net or java applications is a waste of time and money and fits only in a organization that has no focus on quality.**

**The question is not what are the pro's and con's of Uniface regarding .NET and JAVA. The question is, is your approach future proof. what is your return of investment, what is your time to market, what is your software productivity, and does it solve your functional requirements. With those questions in mind, the answer is simple, Uniface can help with high productivity, Uniface is future proof, and Uniface can implement your business requirements.**

## **Ronny Krite**

Uniface Consultant and Internet Application Designer

**A small comment regarding time-to-market...**

**In my experience, Java applications have always been scoped out at a much larger build-to-deployment time than a comparable Uniface scope for the same application/requirements. (FYI, We have had several projects that we scope for a Uniface implementation as well as a Java implemenation.)**

**Another small comment :)....**

**Although Java is free, it is not a cheap alternative in the slightest.**

## **Theo Neeskens**

Competence Leader Uniface at Compuware

### **Complexity:**

**While 3GL's are perfect for writing tools and small applications, you need a 4GL for a large business application. Only a 4GL hides enough technical details from the programmer to focus on delivering business functionality. Java and .Net are not bad tools, but they are the wrong tools.**

### **Productivity:**

**In a traditional client/server architecture you can reach 2-4 hours per function point with Uniface. No mainstream tool can beat that.**

### **Application lifespan**

**Next year Uniface is 25 years old and it never, ever had a technology break. You can still run programs written all that time ago. And Uniface will still be around for a long, long time. Can you imagine .Net or your particular brand of Java still being around in 25 years? And still running the programs that you write today? Without needing a mayor rewrite? And with options to use technologies that are current in the year 2033?**

### **Runtime stability:**

**In my experience the Uniface runtime is VERY stable, and it is even hard for programmers to write programs that crash. Not a small achievement for a multi-platform tool.**

## **François Moritz**

Architect & Uniface specialist & resp.of Standards at Hewitt Associates

**To make the comparison between .NET / JAVA or Uniface calls me. It is only about tools...**

**The way of conceiving an application depends on engineers, on the vision which they have of the real world, the computerized world and the future world. On no account, a tool whatever it is cannot help whoever it is in the obtained result.**

**It is true that at present even too many responsables remain hung on trusting on figures...**

**To want to measure the success among lines of codes, references to objects still stays current for certain Companies and I leave the care to the specialists of the audits to continue in this way because they still have beautiful days in front of them.**

**A tool can never bring more that what the analysts want to set up.**

**I work on Uniface since 1990 and with other tools, I can say to you that I met many applications built with different languages.**

**Some were badly thought, the others thought well from the beginning, only these last ones survived the evolution...**

**In our world, the evolution intervenes every moment. The revolutions succeed one another, the fashions also.**

**To conceive a computer system without having a structured approach limits medium and long-term any possible evolution.**

**For such systems, when the cost of implementing new functionalities exceeds the reason, It is then very often too late to rethink the system, the engendered costs become too consequent, and the system dies. You know the "game of the life" by Johannes von Neumann ?...**

**But why nowadays, still exist applications in COBOL?**

**The choice of a tool stays of the domain of its usefulness.**

**Uniface has weak points like .NET and Java have.**

**I stay of notices that Uniface is flexible for whom knows well how to choose and to use its functions.**

**Of course, Uniface has its limits and to want to go further there it is necessary to know how to choose the good options as in other languages.**

**But its repository leaves free access to the needs of every company. Each can define, adapt his protocols to his methods easily.**

**For my part I am always convinced that the productivity is impressive and that the maintenance remains bound to the made choices of implementation, not the tool.**