'eVo'

..things change.

Software that evolves and lives everywhere

PEOPLE and SOFTWARE - There exists a gap between users and developers. Between the logical perception of a button click and the technical implication underlying it.

What's more is, it's a daunting divide which more often than not requires the participation of specialists. Terms like 'programmer', 'systems analyst' and 'business analyst' pop up.

And then there are aspects such as 'service provider', 'host' and 'compatibility'. Not strictly technical but these phrases seemingly imply the need for technically minded people, people who are difficult to get hold of.

There seems to be an implicit assumption that software solutions don't need to evolve. And once development comes to an end even the slightest change can involve huge amounts of resources. What should have been an instantaneous change is instead scheduled for discussion, experts are called in and meetings are scheduled. Often user desires simply fall by the wayside because quite simply, it's too hard to implement.

What does the past tell us ?

We began by processing ones and zeros. Soon we packed all that into machine instructions and almost simultaneously that was followed by a formalized 'Assembly' language. Finally, the World saw the first formal programming languages – 3GL's. But these to were soon inadequate as they were cumbersome and laborious. There had to be a quicker way to implement logic programmatically. The first of the 'C' variants saw the light, and even then it soon improved to C++. Java made it platform independent.

Then, Steve Jobs gave us our first 'Visual' interfaces when, almost by coincidence, he saw the 'window' icons used by IBM. Bill Gates in turn copied from Steve and by November 1985 the World saw the very first by-and-large graphical user interface. The 3GLs needed to become visual. Microsoft turned 'Basic' into 'VisualBasic'. The owners of 'Pascal' turned their tutoring IDE into 'Delphi'. The likes Fortran and COBOL (which only lived on mainframes) died. Thus, the first truly graphical programming languages had come about in the form of 4GLs. As for the rest, a very competitive market evolved with each coding language accumulating its own parochial following. More and more programmers had access to vast libraries of pre-defined components, no need to re-invent the wheel.

This level, the 5GL era, seems to be the place where the world of software development has stopped.

Each time, with each new level, the gap between low level machine logic and a final user-readable interface became smaller.

Imagine for a second what the world of Information Technology will look like 10 years from today. Will there really still be a horde of technical companies specializing in single aspects of I.T. ? Or will the natural drive toward ease and friendly usability have decreased the gap even further ?

Death of the geek.

Something tells me that we're on the verge of a 6GL.

That would be **eVo**.

eVo allows users to build their own software solutions. **No need for coding, instead you model your**

eVo is not technical, it's easy.

thoughts. Drag-and-drop anything you want, it's all there, on your tool pallet.

eVo allows users to use the pre-built solutions of others, and merge it into their own. Why limit your menus to your own ? Why not interlink with other eVo applications since you share the same interests.

eVo can travel across the Internet and facilitate business partners from all over the world in a real time manner.

It shares data with anyone allowed and gives them a pinpoint place to commence with contributions by the simple click of a hyper link in Google.

eVo puts participants on the same page.

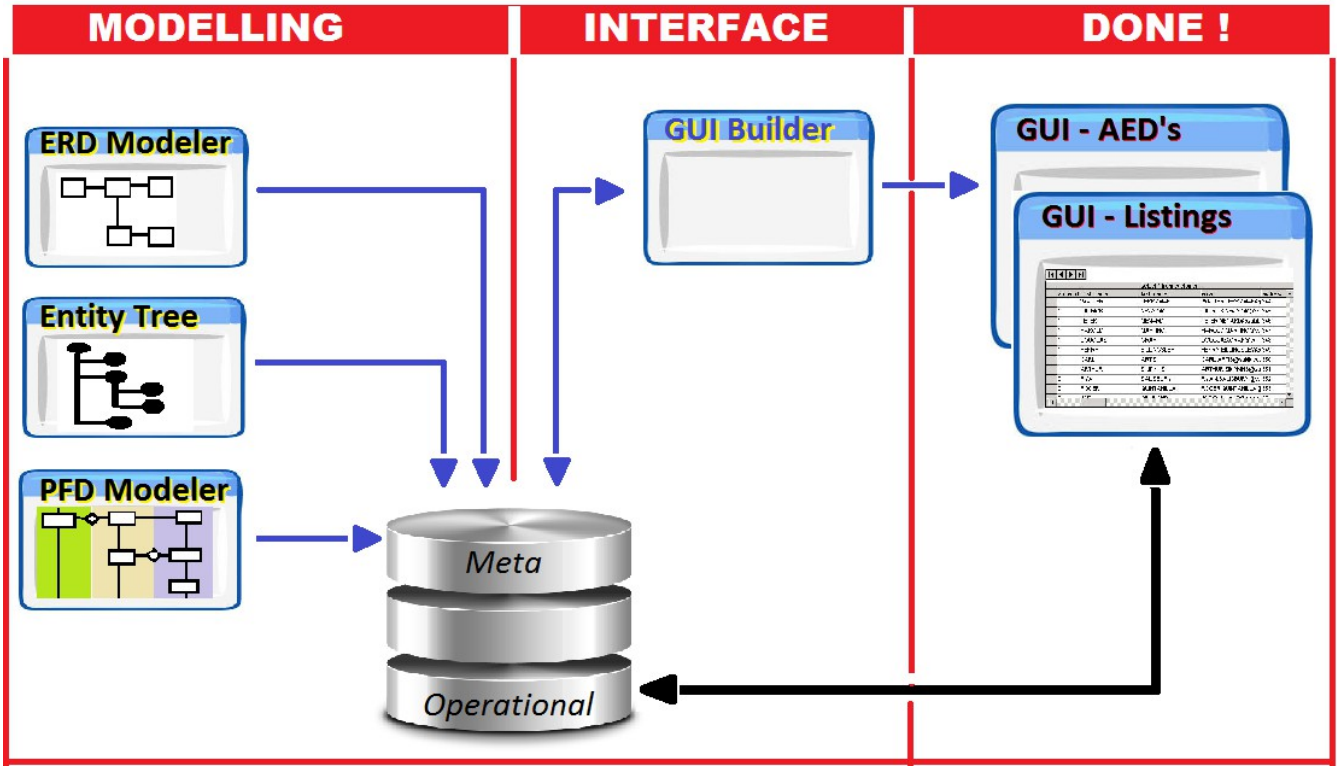
eVo allows users to share computers in the same way that Team Viewer does, it allows Emailing in the same way that Outlook does, it sends text messages like Messenger, it maintains general ledgers and prints invoices, it schedules appointments and manages projects. It does anything anyone wants it to do. And it allows you to have more the next day. eVo evolves with simple clicks of a mouse, not through Systems Development Life Cycles.

Through its shared online library, eVo does all this for free.

How does eVo work ?

eVo

High level PFD



eVo uses similar methods as the popular systems development tools, in that it to models (as opposed to writing code straight away). But there are two main differences :

- **eVo's Models are dynamically linked to the deliverable. Changes can be made at any time to the model and the effect on the operational program is instantaneous for all participants**
- **eVo's model changes have no structural database implications and there is no need for a re-compile of EXEs or a release process.**

The actual eVo executable is in fact a very thin client and will use insignificant space or resources.

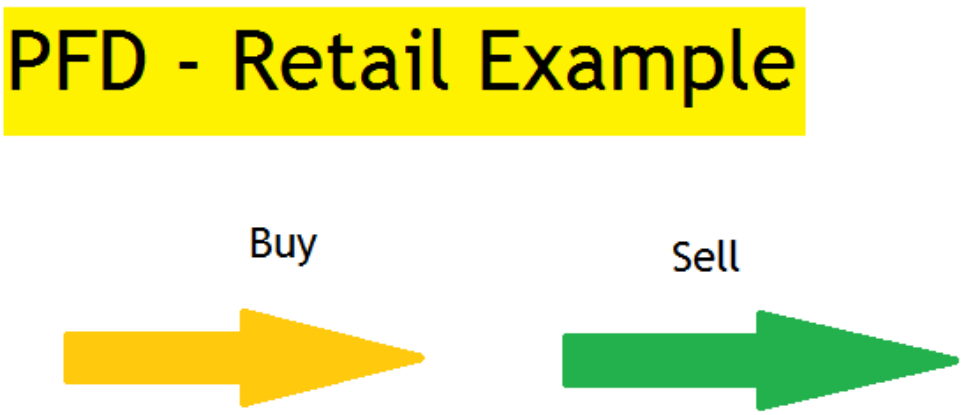
Its internal programmatic logic has an almost linear compatibility with another popular technology – XML. Entire eVo program-data packages will be transported using the XML format and its platform independence is as a result of this compatibility. Whether the end result is to be a desktop application, a web page or a mobile app, the logic stays the same and any project will be 'mobile-ready' at the outset. Want it on the web ? Then put it on the web ! Need a smaller sized version of your program for mobile ? Easy ! allows snippets of both data and software to be extracted and used as required on any OS and

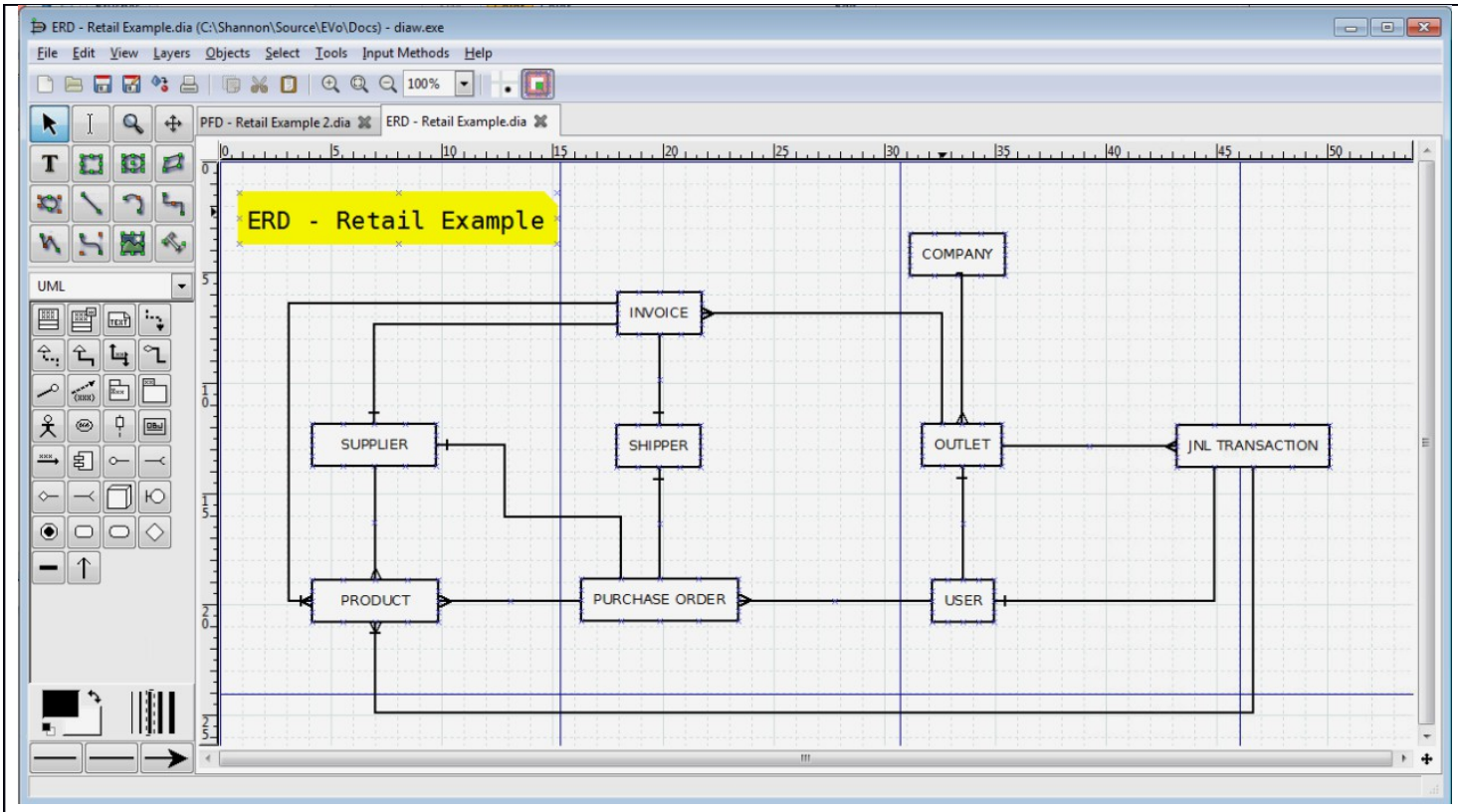
on any device. Many manufacturers are already using eVo on their devices – printers, copiers, scanners, TVs and home appliance. No need to train your universal remote control, there is a custom eVo interface waiting online that will do it for you.

A sample project.

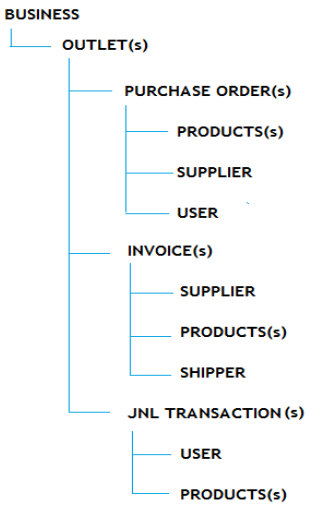
Steven is a year 10 pupil. He has been tasked by his tutor to set up a business plan for a retail business. As part of this project, he is to "develop" a simple software solution with which to track his business activities. His Business Economics class activities and his Science & Computing class activities will therefore overlap – at the one he gets the business acumen for trading while at the other he gets to build the tool to facilitate it with.

He downloads a copy of eVo and opens a new project. The temptation is there to grab a copy of an existing eVo program online – one that someone else has made available one year earlier as part of an existing business project. But he knows he's not allowed to and all eVo projects are stamped to include the details of its creator.

So he starts by thinking logically : "What will I be doing in this business ?" The answer is simple of course – he will be buying and selling. So he starts by recording those two actions.	
Keep in mind that eVo has more than one way of modeling. It all depends on the user's inclination. Firstly there is the E.R.D. (Entity Relationship Diagram) method :	



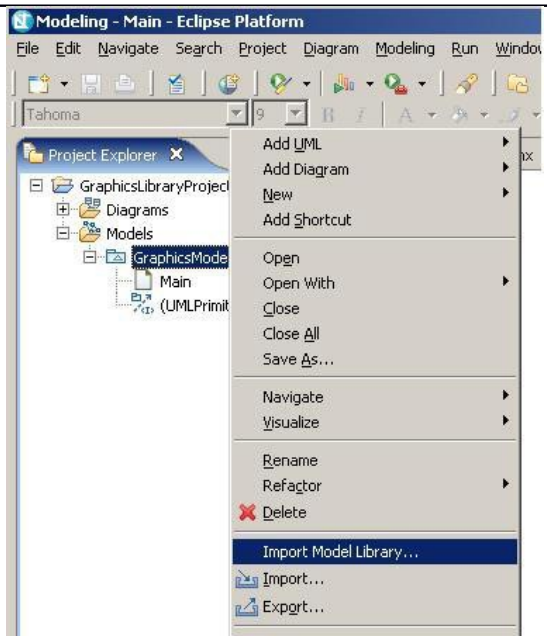
This model is often better to use at the outset as it obligates users take time in thinking about "what" and "who" they will be working with, as opposed to Process Flow modelling which allows the user to think in terms of what activities they will be engaging in. However eVo will allow users to 'discover' entities as they model and incorporate them into the project. In addition, eVo will also think for the user, and based on it's extensive archives of existing templates, will offer to auto generate many elements. Rather defeating the aim of the exercise if Steven is actually meant to think for himself. But fear not, educational configuration within eVo will allow only those parts of the functionality as specified by the administrator.



In addition to the ERD method of modelling eVo also has a simple, to-the-point tree view tool which allows users to see their trading world in terms of hierarchy. Certain things 'own' other things, such as Suppliers who have many Products. A tree view of the above might therefore look something like this.

It doesn't really matter which method is used and it also does not matter as to precise sequences and inclusions. eVo will arrive at basically the same solution. And then of course (one of) its main features comes to the fore – the ability to change both graphical user interface and the underlying business objects as you please. Nothing is unchangeable and what's more is that change is made with the greatest of ease.

Let's assume though that Steven, like most people who are new to

programming, prefers to start with a sequential action driven model – the PFD (Process Flow Diagram). **Next, he likely will have to picture himself or his business in his model.**	
Then, as he thinks about who and what he will be dealing with, he includes some more logical entities in his model :	
As with all true interactive modelling tools, Steven's model is alive in each piece of it. So he clicks in a number of places and checks out the pop-up menus :	
	
Great ! He can add all sorts of things.	
So he decides to insert another entity he thinks he'll be dealing with as well as an action.	
Modelling allows him to think at a high level. No technicalities at all,	

not at this stage anyway.	
During modelling eVo will allow users to give shape to their logical grouping of business functions:	

Finally Steven arrives a model which he thinks is sufficient for now.
At this point 2 important considerations should be mentioned - Database and changes. Systems developers experience a measure of agony as to the adequacy of their design once modelling and design is complete. Although changes can be made, the iterative process of doing so implies a fairly involved process with many considerations - such as backward compatibility. Firstly in terms of the underlying "business objects", secondly in terms of the related interface implications and thirdly in terms of the 'back end' or database – specifically the database structure (or meta data).

For eVo it's all part of the plan. Firstly, changing business objects (via the model) is all done in stride and the effects are seamless and instantaneous. Secondly, there is no worry about the database structure, it's all facilitated automatically and does not require a separate design process. So to for the "GUI". But get this, if a user so wishes, a model for the database can be generated and edited never the less, in the same manner as the ERD.
So there it is. Done ! or is he ?

Well, yes and no. eVo will now, based on Steven's model, generate a fully functional EXE (executable). But the 'GUI Builder' facility of eVo allows him to customize his program a bit further. He can now manipulate his visual components manually in terms of position, size etc. He can also group them logically and add custom menus.

He also decides to include a menu link to the application of his reciprocal team in China who has a similar project.

At this point Steven's teacher allows the class to select one chunk of pre-designed and ready built eVo software to include in their projects. Steven decides to include a piece that processes orders. He gets this from his Chinese partners since the solution of both partners use the same structure for Orders and hopefully their collective bargaining power can secure a better price during purchasing and Steven would like to tap into the same supplier pool as his Chinese partners while at the same time allowing his Chinese partners access to his own customers with a special agreement involving a commission structure.

As for processing deliveries, Steven has decided he wants to implement the programmatic logic himself. He will use eVo's Action Logic tool to spec the procedure for the receiving of orders. In this he has a running start because eVo has already generated all the necessary display screens

for all of his business objects. He has a screens to create new orders, a screen to delete them and a screen to edit them. He decides to make use of a Object Oriented feature to build a screen to process orders with – he "inherits" a screen off of the existing Editing screen and calls it 'Order Processing'. The underlying editing capability for all relevant fields is seamlessly implemented by eVo.

Next year Steven will implement actual code for some of his routines and make his system more specialized. Then, as now, he will have a world wide pool of participants who's logic he can tap into and share.

And there it is, a new application !

Notes of value :

- eVo has a Logic Language facility for specifying precise implementations of actions.
- The facilitation of programming methodologies – OO, patterns, re-usability
- Offers comparative designing
- Users can view actual 'under-the-hood' code at any time.
- Help us lay the foundation for standardized symbology.

- Code generator – C++, Delphi, Java, HTML
- Prototyping tool used as such by system designers