Assima
Innovate & Accelerate

The Future of Software Simulations

A review of the benefits and constraints brought by the successive paradigm shifts, as the industry of Software Training progresses

Author Eric DUNEAU

Date 03/11/2009

Version 1.0

Audience All

Overview This whitepaper reviews the constraints and benefits associated with training options and environments, and the gradual progression towards the full virtualization of training environments.
It also establishes Assima's contribution to the challenge, highlighting some unique benefits of Software Clones, and Virtual Clones.

Table of Contents

It all started with a Training Client

Identifying the most important risk with any new software rollout

A company-wide software rollout, even if it is often a large accomplishment in itself, is probably only the tip of the iceberg. The iceberg represents the company's capabilities to deliver value and growth in a predictable and standardized way. The 10% tip is the Information System which helps standardize processes and raise efficiency. But as we know, a system is only as good as what its users will do with it. And most of that is hidden below the waterline.

If a system is not used, or badly used, well… it ain't gonna deliver its promises (of standardized processes and better efficiency). So that is why the most important risk with any system rollout is the level of user acceptance.

Why is a training environment a necessity?

How do you get good end-user adoption when you roll-out a new system? By demystifying the system, making clear why it is being deployed and what it shall do (information), and making clear to its future users how it works (training).

A good training environment (simulating the production environment) is therefore one of the most important environments to setup, alongside the production system itself, so that end-users spend some time in it, learn how to use it, and get rid of any fears that could endanger good use of the real system.

It is therefore extremely important that the training environment is as close as possible to the real production environment in terms of use (not just look and feel, but actual use). That is why a Training Sandbox (a copy of the real application, on a non production database, with some well defined data scenario) has historically been a standard approach in terms of end-user training.

Recurring risks and costs of a Training Client

Despite being probably the best training solution, the Training Client comes with a long list of risks and costs. The most important ones are:

Risks:

- Does not work if the network is down
- Does not work if the database is not refreshed
- Depends too much on the physical infrastructure (availability of classrooms, etc…)
- Does not scale

Costs:

- Requires heavy maintenance (resetting the database, upgrading software, upgrading storyboards)
- Requires heavy infrastructure (database, software, hardware, middleware and network…)
- Requires taking trainees off site (off workplace) for long periods

Quick review of the qualities of Training Clients

To understand better the concepts at stake, we can use a simple analogy. Let's say that the production environment is a Boeing 737, and that your goal is to teach several of your employees how to fly it.

Additionally, to reflect real world experience, let's also say that just after your first training environment has been delivered, the real Boeing's altimeter has had to change following new regulations. This change has to be passed onto the training environment, and you have to evaluate the impacts.

In our example, the best representation of the training client is a Flight Simulator with the full cockpit, knobs, software, and transmitted vibrations into the cockpit.

The realism of the training environment is very high.

Now, if you want to reflect changes appearing in the real Boeing cockpit, you have to make changes to the hardware, software, data, on all Flight Simulator instances (that could be very costly).

Our altimeter has to be replaced (hardware, software, cables, etc…) in all the Flight Simulator cockpits.

The Training Client is your Flight Simulator

Type of Environment	Physical copy of the reality
Realism	★★★★★ Real data, real logic
Benefits	High retention rate, high end user adoption
Drawbacks	Expensive, risky, complex, not scalable

The undelivered promise of Simulation Tools

The promise of eLearning

The arrival of eLearning (in short, training content and training platforms accessible over internet/intranet) has created a big paradigm shift, with the promise of new benefits. Trainees would not need to travel, content can be produced at a low cost and by non technical experts, the scalability of the solution is high, the traceability is total, and so on… On paper, it looks like it's a very progressive paradigm shift. It does indeed deliver on this promise for certain types of content and infrastructure, but it does not for the particular case of systems training.

For systems training to be efficient and drive high end-user adoption, it needs to provide total realism. This is the big drawback of eLearning simulations, as they are extremely static and unlike the real application they are supposed to simulate.

Therefore, despite all the benefits brought by the paradigm shift, in the particular case of systems training, the switch to eLearning tends to bring a retrograde solution.

Simulation Tools and the screen copy boundaries

Simulation Tools have evolved from being extremely static solutions (video, point & click screen capture) to a slightly richer realism thanks to the addition of data input, and multiple path branching. But before we expose the limitations, let's review how they work. All Simulation Tools work on the same principle:

1. A capture tool, which, for each end-user action in the real application:
    - takes a screen copy of the application
    - captures additional information live on the application, and enriches the screen copy with this information (such as edit fields, hotspots…)
    - stores all (screen copy and information) in files or databases
2. A simulator, which can playback the screen copies and additional controls on top of them (generally hot spots and edit fields)
3. An editor, which can edit:
    - how screen copies are chained together
    - some properties of the additional interactive controls (size of hotspots, content of edit fields)
    - additional instructional text placed on top of the simulation, for the benefit of the end-user

The quality of a Simulation Tool depends strongly on the realism of the output, which in turn shall depend mostly on the type of additional information provided on top of the screen copy. However, the screen copy itself acts as a barrier to the integration of complex controls which would improve the realism, as it displays a static background image which cannot change in synchronization with the control behaviour. It is therefore not possible to simulate, with a good realism, roll over effects, selection effects, scrolling effects, drag and drop effects, resizing effects, treeviews, menus, grids, and the list goes on… That is why most Simulation Tools just implement basic controls such as hotspots, edit fields, combo boxes, possibly

buttons. That limits greatly the type of interactivity, multiple paths, and the resulting realism. All screen copy based Simulation Tools are limited in their capacity by the screen copy boundary. No exception.

Maintainability of content in a permanent Change environment

In addition to the problem of a lack of realism, Simulation Tools carry another heavy burden, also a consequence of the screen copy technology at the heart of the solution. As production systems change every month, quarter, semester, the content produced by the Simulation Tools is locked in the past, with very little possibility of upgrading it easily. It is totally impractical to repaint all screen copies. The only solution is to redo the job (re-capture, re-edit, re-test, re-etc…). Every month, quarter, or semester! It makes the solution totally impractical and not adapted to the requirements set by Systems Training.

The lack of maintainability of content is probably the most critical problem associated with screen copy based Simulation Tools, as it confronts the needs of permanent content upgrades, or a risk of permanent content de-synchronization with the production environment. The promise of a low cost content production solution takes a beating. The reality says there are very high maintenance costs attached, probably un-affordable on a large scale project.

If you add the lack of realism (low end-user adoption) to the high cost of maintenance, the natural conclusion is that Simulation Tools provide very little benefit in Systems Training.

The repositioning of Simulation Tools

The market has progressively taken the measure of the qualities and constraints of the Simulation Tools. In 2000, there were probably more than 20 independent players on this market (OnDemand, RWD, STT, Firefly, IBM Simpro, Inovae, InsiteObjects, Knowledge Solutions, x.Help, SAP Tutor, Datango, Team Trainer, XStream, Quarbon, RoboDemo, OutStart SoftSim, Toolbook AuthorWare, Camtasia, …). By 2010, considering that SAP and Oracle have acquired their solution, there will probably be less than three independent players, dominated by Adobe Captivate, which is gradually implementing most functionalities of the past competitors, for $500. The pressure on the last two means they will probably disappear soon, allowing this market to reconfigure around:

- Oracle with its own proprietary solution,
- SAP with its (probably soon) own proprietary solution
- Adobe for the mass market

Simulation Tools are therefore becoming commodities, and as the cost of ownership is lowered to the very minimum, their audience widens, the needs coverage explodes, and the positioning slowly drifts towards multimedia and rapid learning solutions: quick and cheap tactical solutions, as opposed to strategic and cost effective long term infrastructure solutions. It's a bit like promoting Excel in place of an ERP. On paper, it can do the job with more and more Excel sheets and macros. Practically, it's a waste of effort.

In conclusion, it is the drive towards lowering training content production costs that created the space of Simulation Tools ten years ago, and it is still the same drive which keeps the illusion alive today. Simulation type content creation is cheap, Simulation Tools even cheaper, and ever more accessible. However, as the mass market widens, it is unlikely that the Simulation Tools provider(s) will ever

innovate and deliver truly adapted solutions for the Systems Training niche market. Although it is cheap, it is not fit for purpose, and will probably never be.

Quick review of the qualities of Simulations

The simulation is nothing more than an analogue copy of the reality (a picture, or a film). This picture or film can be enhanced from time to time with special effects, but will never come close to delivering a proper training experience.

The realism of the training environment is very poor.

Now, if your real Boeing cockpit changes, you have to retake all the pictures, the movies, and redo all the special effects, as your analogue simulation cannot be re-edited easily.

All pictures and videos where the altimeter is displayed have to be retaken, and as the new picture is re-taken, all special effects that were made before have to be redone, even if they do not apply to the altimeter itself.

The Simulation is a picture of your flight simulator

Type of Environment	Analogue copy of the reality (film, picture)
Realism	★☆☆☆☆☆ Static data, static logic
Benefits	Quickly produced, fully scalable, traceable, accessible from the web
Drawbacks	Not fit for purpose, low end-user adoption returns, highly un-maintainable

Beyond the screenshot boundary: Software Cloning

What is Software Cloning?

Although it addresses the same challenge of how to train end-users on IT Systems efficiently (adoption) and at low cost (maintainability of content), Software Cloning is very different from the solution provided by Simulation Tools. It does not make use of screen copies. This is the only major difference, but it is big enough to provoke another paradigm shift, this time a progressive one.

The three major components (capture tool, simulator, and editor) are still here, and provide equivalent functions, albeit very much improved.

The capture tool will only capture information from the original application (no screen copy capture). This information is generally made up of two types: static information (objects, their properties and values), and dynamic information (actions on objects, and their effect on other objects).

The simulator is what creates the end-user impression, and the realism. This realism depends very much on the quality of the information captured (at the smallest level of detail), and the quality of the simulator to playback this in-depth information. A powerful simulator implements all the look and feel and behaviour of the graphical components which make up an application, down to the smallest detail. A powerful simulator will render the clone realism so close to the one of the original application, that the end-user will see no or very little difference.

As the information captured is structured information, and not a screen copy, the editor can allow the change of any piece of information, thereby changing accordingly and instantaneously the look and feel or behaviour of the clone.

End User and project perspective

As the clone is comprised only of graphical objects, the end-user will enjoy a training experience which is closer to the one delivered with a Training Client. The majority of the application's interactivity and multi-navigation possibilities are recreated. The main difference with the Training Client lies mostly in the application logic which is simulated by the clone with one dataset, but not recreated (no capability to apply the logic to any dataset).

Structured information also allows rapid edition, easier maintenance, and additional benefits such as swapping text by an alternative (localization) or swapping values (data protection) via easily maintained dictionaries. It gives the project owners a lot more flexibility in keeping all clones up to date with the application, and these benefits are multiplied in complex multilingual, multicultural environments.

Clone vs Object based capture, what's the difference?

As far as marketing goes, everything looks the same in the world of object based capture. And it is indeed quite easy to get confused, and not see any difference. The capture technology used by most (if not all) Simulation Tools works on the basis of screen capture and additional object recognition capabilities. Clone capture works on the basis of object recognition. So what is the difference, and if there is any, and is it that important?

Yes, it is very important! The difference is subtle, but is important enough to trigger the paradigm shift.

For a clone to exist and be simulated properly, it requires 100% of the target application interface to be captured, in all its tiny details. If a single property of an object is not captured well (background colour, position, z-order, vertical alignment, virtual key, or any of the hundreds of properties for each object type), then parts of the clone will not work as expected. Any defect in the capability of the capture tool, or the simulator is highly visible in the simulated output, and creates a gap in the clone.

For a screen-shot based simulation to exist, it just needs to capture an image in the background. All other object based recognitions are a bonus to add interactivity on top of the screen copy. And they can be very partially captured, such as the object position and size only for a hotspot.

So there is a world of difference between on one side capturing the complete set of graphical objects and properties and actions, and on the other side capturing a few objects and properties here and there. One allows an easy edition of the structured data to remodel the clone UI (redesigning the forms and graphical controls, localizing the screen UI, protecting data, etc…), the other one is stuck for ever within the limitations of the screen copy (any change in UI or data in the production environment triggers a full recapture of the simulation, and a full rework).

Furthermore, there are other complications in the quest for cloning applications that go beyond the capabilities of common object based Simulation Tools. Let's review two cases showing the significance and value of the object capture tool, and the simulator tool.

- Web based applications unfortunately do not have a rich list of controls at their disposal, as these controls simply do not exist in HTML. That is the case for treeviews, tab controls, grids, listviews, toolbars... It means that any HTML application trying to display information structured as a treeview or a grid, has to use more basic HTML components to do so. The most sophisticated object based Simulation Tool would capture the basic HTML objects making a treeview and would totally miss the point, ending up with irrelevant information, and not the treeview object itself. The value of the capturing tool is not only of capturing objects and properties, but of capturing the visually intended object and associated properties (in our case, the captured treeview object was never a treeview object in the real HTML application).
- Some Java, Windows or .NET based applications tend to give developers more liberty around the development of custom graphical controls. The trend is set for the creation of new controls, or the creation of commonly known controls with new functionalities. For example, a grid control could implement column selection, whereas it is not a commonly accepted feature of a grid. The richness of a simulator is its ability to simulate commonly accepted object behaviours, but also rarer features that contributes to the quality of the original application.

In conclusion, there is indeed a world of difference between object based simulation technology, and software cloning technology. The capabilities do not come close, for the capture tool, or for the simulator, and therefore the induced benefits of realism and easy maintenance do not come close either. So long as Simulation Tools continue to use a screen copy in the background of their simulation, they will not make the jump to the next paradigm shift and its associated benefits. And that will still hold true even with more powerful object recognition. The screen copy acts as a barrier to better interactivity and better maintainability of content.

Quick review of the qualities of Clones

The clone is a digital copy of the reality, made of editable structured information which is played by a robust application simulator.

The realism of the training environment is high.

Now, if your real Boeing cockpit changes, you can update the information contained in the clone, and change it. You can do partial maintenance automatically (data protection, localization of interface), and other parts manually (re-design to reflect the change in the original software).

You can edit and redesign all clones containing the altimeter, and change the look and feel, and behaviour of the altimeter. All other behaviours in the clone do not require updating and stay untouched.

The Clone is a digital replica of the flight simulator

Type of Environment	Digital copy of the reality
Realism	★★★☆☆ Structured data, navigational logic
Benefits	Good end-user experience (adoption), good maintainability, fully scalable, traceable, accessible from the web
Drawbacks	Requires a setup phase

The virtualization of a Software Clone

Still more value can be delivered

Since its creation, Assima has always worked with a dual goal in mind:

- Delivering what is best for the end-user
- Delivering what is best for the project

Assima argues that the most important criteria for the end-user is that the training environment is as close as possible in terms of realism as the production environment, but at the same time, is available 24/7, on demand, scalable, traceable, and just a few mouse clicks away at the workplace or even from home.

Assima also argues that the biggest challenge ahead from a project perspective, is to maintain the synchronization between training content and the live application in a timely manner. Following the flow of upgrades to the live system is no easy task. It requires being able to update potentially thousands of training materials within a very short space of time. It is just not feasible at all with Simulation Tools, because of the limitations set by the screen copy, and can become a challenge even with software clones as the volume of change increases.

Assima is launching new R&D initiatives to push further the value created by the Clones, both for the end-user, and for the project. This work will lead to prototypes in the second half of 2010, and an official product in 2011.

Enhancing the clone's realism

There are ways to enhance the realism of the clone, and it mostly does not require any new technology or new paradigm shift, but the continual improvement of the capabilities of both the capture tool and the simulator. Ideally, not only the objects can be fully cloned (properties and behaviours), not only the multi-navigation options of the original application can be recorded and played back, but some of the application logic could be introduced into the clone, and played back by the simulator. If the structured information captured and stored in the clone could simulate more application logic, then the simulator could playback the clone with more datasets, rendering the end-user experience even more realistic.

Enhancing the clone's maintainability

Improving the maintainability of the clone, and shortening the update cycle, however, requires a bigger step forward, a new progressive paradigm shift. This step forward, built on software cloning technology, takes the structured information captured to the next level, where the modification of an object's property in a screen gets propagated instantly and automatically to all training content in which this screen is embedded. The update is made once only, the first time it is seen in any context, by editing the structured information (edit properties, values, actions…), or by recapturing if changes are significant. All content is then automatically updated with the change. When implemented, this technology will deliver easy maintenance of huge volumes of content at very little cost.

Quick review of the qualities of virtual clones

The virtual clone is a virtual copy of the reality, which integrates one major benefit of the virtualization: speed of adaptation.

The realism of the training environment is high.

If your real Boeing cockpit changes, you can update the information contained in one virtual clone, and change it. All other virtual clones using this same information are automatically updated.

You can edit and redesign the altimeter cloned object itself (look and feel and behaviour) independently of any clone using it, and all virtual clones embedding such object are automatically upgraded with the new altimeter.

The Virtual Clone updates itself automatically

in synchronization with other virtual clones updates

Type of Environment	Virtual copy of the reality
Realism	★★★★☆ Smart data, simulated logic
Benefits	Highly maintainable, truly life-like, highly scalable, traceable…
Drawbacks	Requires a setup phase

A summary of all options and associated constraints and benefits

The table below summarizes the options available for systems training, and the various paradigm shifts existing between options. The search for the right solution is often a compromise between the quality of training and resulting end user adoption (realism) and the cost of the system (cost of initial setup, cost of maintenance).

For their intrinsic characteristics (scalability, ubiquity, traceability, and more…), the solutions coming from the Simulated Worlds will ultimately prevail. However, as Simulation Tools represent a significant step back from the Real World, only Software Clone based solutions can deliver the expectations set by the needs of large IT systems training. Virtual Clones, or Application Avatars, will take the virtualization of the training environment one big step further, with a promise of low cost maintenance on all content upgrades.

For the past ten years, Assima has engaged in massive R&D efforts in making Software Clones, allowing the first progressive paradigm shift of the Simulated Worlds. Further efforts are now being spent to turn Software Clones into Virtual Clones, or Application Avatars. This second progressive paradigm shift brings a new value proposition, re-conciliating the conflicting interests that the business has always had to ponder, between what's good for the end-users (better realism for better system adoption) and what's good for the project (lower ongoing maintenance costs).

	Real World	Simulated Worlds		
		First (retrograde) paradigm shift	Second (progressive) paradigm shift	Third (progressive) paradigm shift
	Training Client	**Simulation**	**Clone**	**Avatar**
Solutions	Hardware + Software + Database	UPK/uPerform/Datango/Captivate	Assima ATS	Assima AIS (2011)
Environment	Reality	Analog copy	Digital copy	Virtual copy
Realism	Real Data Real Logic	Static Data Static Logic	Structured Data Navigational Logic	Smart Data Simulated Logic
Project view	High cost of setup High cost of maintenance	High cost of maintenance Limitation in output quality Cannot do complex projects	Requires a setup phase	Requires a setup phase
End User view	In line with production env. Fear of breaking the system Practice in a life-like env. Highest retention rate	Fairly rapid production capability Traceable Scalable Very frustrating and limited user experience Provides less than 15% retention Always available Standardized training	Same benefits as Simulation Better maintainability of content Generate content from one src - multiple output modes - multiple languages Ease of redesigning (data, logic) Same benefits as Simulation Good end user experience High retention rate	Same benefits as Clone Magnified maintainability One change updates all project Same benefits as Clone Even more life-like experience
Business view	Too expensive Too risky Too complex (skills required) Real training env. (quality)	Un-maintainable, quickly outdated and low retention makes it a wasted investment Allows savings vs Training Client	Same benefits as Simulation	Same benefits as Clone Lower maintenance cost