Rapid Application Development [RAD]

RAD In Solution Delivery, Beyond Prototypes

lyte³

www.lytecube.com

Author:
Girish Bellalcheru
CTO, Lytecube.com

Abstract

Rapid application development has traditionally been used in building prototypes, There are not many success stories of using RAD tools in building complete solutions because of their inherent limitations. This paper looks at how RAD tools could use a more holistic approach to build solutions rather than being a code generation technology. And finally a few benefits of using LyteRAD a new generation RAD framework for database technology.

Rapid App Development
Friend or foe

The world needs faster ways of doing things. Machines crunch numbers faster, hard disks spin faster, networks transport bits faster, and the real bottleneck seems to be creating software faster. Development teams are trying out innovative techniques to get new products into consumer's hands quickly. We are limited by how fast we can code. No matter how many code generation tools are available, the developer community is so obsessed with knowing every line of code that was ever written. That is our real speed bump, the human mind can process only so much information and assistive tools are a necessity to move faster.

One of the keys to successful Rapid Application Development (RAD) in solution delivery is innovative tools. To deliver a solution fast enough, we need tools to analyze, design and build systems quickly.

RAD tools have always been built on the assumption that the tool is just an assistive technology to developers and generated code should be editable. That defeats the purpose of rapid application development, It's like assigning the human mind an extra task of trying to make sense of alien code and then modify it. A better approach is to generate components that can be treated as individual building blocks, and not chunks of code that needs editing. Better still the whole solution should be as easily editable and creatable.

Various attempts have been made in the online world in the guise of mashables and web 2.0 content. But the real power of RAD will be when you can build solutions to everyday problems.

RAD and the software models

Software development methodologies can be categorized into pipeline or waterfall like models and evolutionary models. Today evolutionary approaches like prototype, spiral and scrum models dominate the landscape. The need to constantly adapt to changing requirements and clarify requirements on the go make the later models more attractive.

Building prototypes is a well accepted practice in evolutionary models. They help clarify requirements and sometimes validate uses cases. Many times prototypes are built to be thrown away, which makes sense in some cases, but largely building prototypes as extensible pieces that can later become a complete solution is where the most payoff lies. And it is here that RAD tools play an important role.

Prototyping the future

When customers see a well developed prototype they easily mistake it to be a working alpha or beta solution. And there is no reason why a well developed prototype should not be be ready to use. There are many advantages in building strong usable prototypes. It saves a lot of time in re-implementing the actual solution. Developers resist writing strong extensible prototypes because of the throw away factor, and the throw away factor comes in because of the non-extensibility of code over a few iterations. Its like a chicken or egg problem.

RAD tools as component frameworks

If RAD tools can provide a set of reusable components that can be readily used in a solution, it can dramatically enhance productivity and reduce development time. An big advantage of using such component frameworks is that you get to test a solution as you build it, which is a dream come true for many developers. It is obvious that code cannot be completely validated until it is fully functional. But with the components approach, each block is fully functional by itself making it easy to test drive the solution as it is built. Starting with something that already works, developers reduce development time. Imagine adding a forms component, and then immediately adding a few records, or adding a reports component and

verifying what it generates. You could add and try out components on the fly and end up with a complete and fully validated solution within a few iterations.

In a typical database driven solution, this approach would shift the emphasis from coding to better understanding of the problem domain. Designing the right solution would be more about identifying the data domain, designing schema and charts to be reusable.
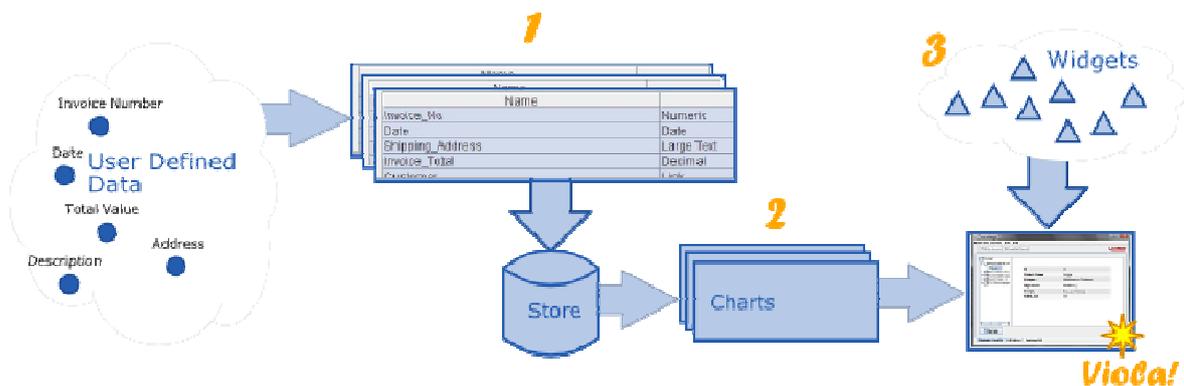

Figure 1

Building the solution would be more about choosing the right components. By using components from a RAD framework, developers can be assured that the code is tested, validated and ready for use on the first day or click, and the solution can definetely be extended over iterations by adding or removing components. The big catch is that the component palette should be complete and ready to use.

Redefining productivity

In the RAD world, Productivity can no longer be measured in lines of code, it would be more like features per hour or applications per week. Even RAD sounds out of place, The right term would be Rapid Component Integration.

From simplicity comes creativity

With that kind of simple tools to build solutions, we can start putting together custom solutions to every little problem we come across. The day is not far when packaged software would be considered an alternative to DIY solutions and not the other way round.

It is exactly this line of thought that has gone into the making of lyteRAD, which addresses the need for DIY database solutions. Click select and build complete solutions in minutes, you can always add components or modify the solution with zero coding. That is when programmer productivity is measured in features per hour. Or TPS as the lyteRAD team wants to put it.

Delivering Solutions using LyteRAD

LyteRAD is a lightweight database component framework to quickly build complete database applications without writing any code. The framework has a base runtime which provides the runtime environment for the components. The component palette addresses commonly used features of database driven solution, like creating tables, data forms, reports charts and a few other analysis tools.
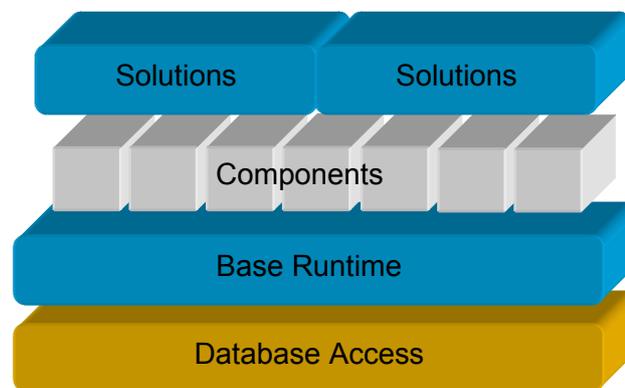
The IDE is designed to mimic the typical design flow of a database solution. As shown in _figure 1_, The tool allows the solution designer to create tables and relationships which will later be utilised by charts and reports. Each component provides a specific feature of a solution like the ability to add records to a table or a visual graph of table data. The final solution is a collection of components and widgets which make use of these data sources.

Adding new features to a solution is as simple as adding a new component to the solution. Component reusability is enabled by the use of a base runtime. The added advantage of a runtime is that any defect fixes and enhancements to a component are available to all solutions immediately.

Lytecube is the maker of LyteRAD a new generation database RAD tool. The company also provides a gallery of ready to use solutions built using the framework.

For more information visit www.lytecube.com