An Example of Old vs. New

To best illustrate the difference between conventional development methods and BriteWorks™, let us take an example of building a simple web application.

Old – Conventional Methods - The development cycle and decision making process has been broken up into several sub-headings for ease of reference. The following are amongst the decisions and steps required	**New – The BriteWorks™ Method -** The following is the same list, however with a much simplified process:
❖ **Environment** ✓ What platform (hardware) will the new application run on? ✓ What database will the application run on? ✓ What are the implications of the above? ✓ What development environment will be used – today, typical Java or dotNET. ✓ For the purpose of this example, let's assume Java is selected.	❖ **Environment** ✓ No early decision making is required ✓ BriteWorks™ is Java based, so it will run on all platforms and databases
❖ **Recruitment of Developers** ✓ Next is the arduous task of finding a good team of developers, let's say five good Java programmers have to be recruited. Because it's a web application, they may need skills in JSF, JSP and possibly AJAX and Java Scripts. ✓ Java developers are hard to come by. There is a regional shortage of these resources and good programmers are expensive and job hop. ✓ There is no point in taking graduates and training them in Java because it will take a long time before they are productive.	❖ **Recruitment of Developers** ✓ No need for highly experienced developers ✓ You can employ graduates with very little experience, one or at most two instead of the five experienced Java developers ✓ It will take three days to train any IT person to user BriteWorks™

❖ **Architecture** ✓ Let us assume that the end user requirements have already been captured, so the next main step is to build an entire architecture, ✓ Building an architecture typically constitutes 30% to 40% of an application ✓ Very detailed technical knowledge is required to build a good architecture ✓ A bad architecture will make the whole system collapse ✓ An architecture diagram has to be created ✓ Architectural details and interfaces have to be disseminated to the developers ✓ Full Documentation has to be done for future reference	❖ **Architecture** ✓ No Architecture Design is required ✓ BriteWorks™ architecture is entirely reusable
❖ **Design** ✓ Each window/program has to be designed ✓ The system flow has to be pinned down ✓ Reviews have to be conducted amongst the development team ✓ Full Documentation	❖ **Design** ✓ If the requirements are captured and a data model is developed, developers can start creating fully functional windows literally within seconds. ✓ Flows can be put together very easily
❖ **Prototyping** ✓ This step allows the user to see an early version of a solution. ✓ Typically, this is done in a tool such as Microsoft Visio, which means it has to be thrown away and has to be redone during the next stage ✓ This could take several weeks depending on the size of the project.	❖ **Prototyping** ✓ The design step above basically constitutes the prototyping. ✓ The prototypes need not be thrown away. They form part of the system as they are fully functional windows.

❖ **Coding** ✓ This is the most complicated portion of development ✓ The least business oriented ✓ The most costly and time consuming ✓ Potentially hundreds of thousands or even millions of lines of code have to be crafted. Sometimes only the developer understands the code written. ✓ Some development teams conduct Code Reviews ✓ Full Documentation	❖ **Coding** ✓ No Coding Required ✓ The system is self documenting
❖ **Quality Assurance** ✓ Integration Testing ✓ System Testing ✓ Functional Testing ✓ Performance Testing ✓ User Reviews ✓ Bug Fixes (by developers) ✓ Performance Tuning (by developers)	❖ **Quality Assurance** ✓ Integration Testing – much smaller scale ✓ System Testing – much smaller scale ✓ Functional Testing – still required ✓ Performance Testing – much smaller scale ✓ User Reviews – still required ✓ Bug Fixes (by developers) – much smaller scale ✓ Performance Tuning (by developers) – much smaller scale

❖ **Training / Acceptance / Migration / Move to Production** ✓ Migration Plans have to be written ✓ Code and databases have to be migrated and kept in synch ✓ Final Developer Testing ✓ Training ✓ User Acceptance Testing ✓ Fixing/Testing (by developers) – followed by new migration.	❖ **Training / Acceptance / Migration / Move to Production** ✓ Migration Plans have to be written – still required ✓ Code and databases have to be migrated – ONLY the repository is migrated ✓ Training – still required ✓ User Acceptance Testing – still required ✓ Fixing/Testing (by developers) – followed by new migration. Smaller scale
❖ **Additional Features** ✓ Desktop vs. Web – in conventional methods, you have to do two sets of developments, so you almost double the time spent. ✓ Security, Auditing, Localization, Import/Export facilities, Preferences	❖ **Additional Features** ✓ Desktop vs. Web – with BriteWorks™, you develop once and deploy in both Desktop and Web, without additional work. ✓ The BriteWorks™ runtime comes ready made out of the box with: Security, Auditing, and Localization, Import/Export facilities, Preferences, etc.