

Business FAQs

1. Are these the reality of your business application development challenges?

1. My projects are over budget and behind schedule.
2. The requirements keep changing and I cannot catch up.
3. I need more re-use across my applications.
4. I am spending thousands of dollars on architecture every day, working on a new one for each new project.
5. I need to start design from ground zero for every project.
6. Different team members write different “flavors” of codes and maintenance is a nightmare.
7. There are a lot of duplications in a project, for example, few people are writing their own audit trail component in a single project.
8. Latest codes are being overwritten due to silly mistakes; this resulted in a lot of reworks.

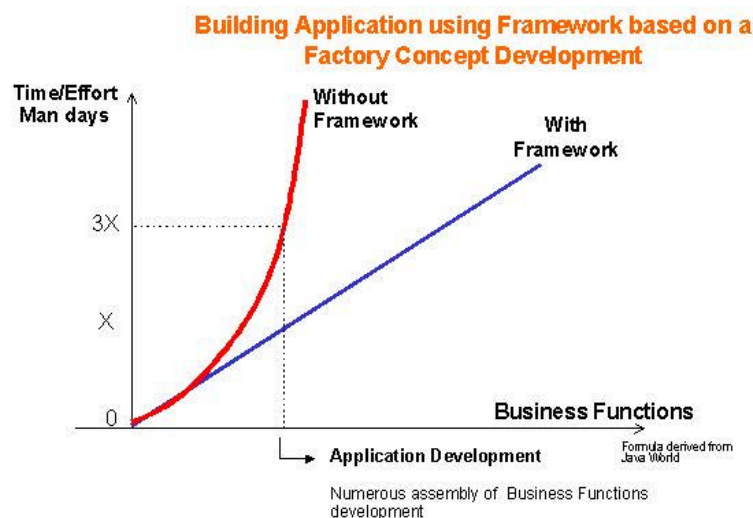
2. How can Kernl speed up design and development?

Design and development effort can be improved due to the following features of Kernl:

1. A fast and easy component-based development platform with streamlined and standardized component integration approach. Kernl (especially with its eDesigner tool) simplifies EJB component development.
2. A set of ready core infrastructure components that can be reused in multiple projects, enabling a divide and conquer approach to coding (parallel development), thereby reducing development time and cost.
3. A factory approach (with a streamlined and standardized programming model) to developing the application components. This further facilitates efficient parallel development.
4. A streamlined maintenance approach, where changes can be made with ease.

3. What is the benefit of using a framework?

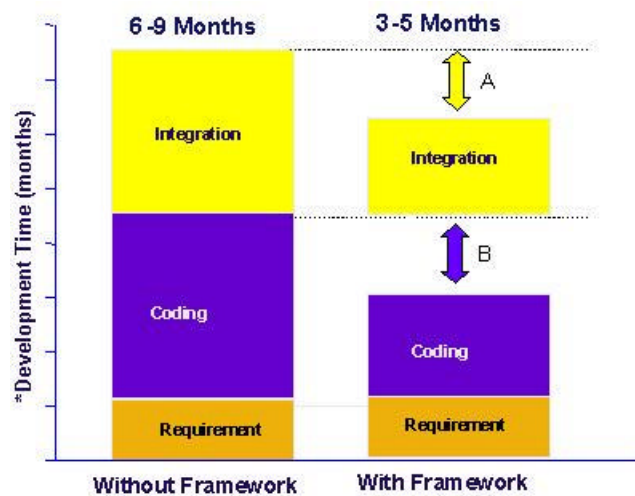
The graphs below show a typical trend in applications development with a framework based on a Factory concept, compared to development without a framework. This is based on research described in Java World (http://www.javaworld.com/javaworld/jw-09-2000/jw-0929-ejbframe_p.html).



As the size and complexity of a project rises, effort rises exponentially due to the large number of factors that can contribute to this size and complexity. Frameworks are designed to linearize this trend

by reducing or keeping to manageable constants the number of these contributing factors. These factors are actually variables like programmer skill, complexity of the problem, etc. Take programmer skill for instance. Because of a typical framework's (e.g. Kernl's) streamlined programming model and facilitation of the factory mode of development, the varying skills of programmers will not impact the quality or development time significantly. This further serves to facilitate more accurate project scheduling estimates.

The next diagram depicts savings in development time for a specific size project that would normally take 6-9 months without a framework. The bar chart depicts savings in development time for both integration ("A") and coding ("B"). Because frameworks streamline and standardize the development process there is significant savings in coding time. And because frameworks also streamline and standardize component integration and communication, there are further savings in development time for the integration effort.



Statistics from Java World report

4. Return On Investment Calculation

A simple ROI calculation can be performed, looking only at savings in development for investment in the developer licence fees. The formula used is:

$$\frac{TB(in\$)\times 100}{TI} = ROI\%$$

where Total Benefit (TB) is the difference between the benefit achieved through the implementation of a Kernl product or service, versus either not implementing the product or service at all or implementing it in a different way. TB can be attained either directly, in terms of incremental revenue gained or expense saved, or indirectly, from the resources freed up for redeployment to other tasks within the organization, hence avoiding the need for new hires or purchases.

Total Investment (TI) is the difference between the total cost and expenses associated with using Kernl and similar costs associated with the most likely alternative approach (which can be none at all).

To simplify analysis, let's assume a conservative estimate of 10% cost savings on design and development effort that would have been taken by a team of 8 developers over a period of 3 years. At a rate of US\$3,000 per developer, the total cost would have been US\$864,000. Taking the conservative 10% overall cost savings, this works out to be a value of TB = US\$86,400. Total Investment (TI) for the developer license is US\$3,500 × 8 = \$28,000. The ROI would be:

$$ROI = \frac{86400 \times 100}{28000} = 300\%$$

Note that by being conservative, we mitigate the simplifying approach taken above.

5. Bean's Experience

Beans Factory's experience suggests that cost savings in design and development is 20-70% when adopting Kernl framework in projects with 3 or more people lasting for 6 months or more.

6. Key Impact on Our Client

1. There was a reduction in development time compared to having no framework and foundation infrastructure components such as those found in the Kernl.
2. ROI of 300% or more over 3 years is possible.
3. Considering the complexity of ERP software, further cost savings may result with increased productivity.
4. Productivity improvements of greater than 50% may be achievable on large projects as developers concentrate mainly on business logic developing Java Classes (and not EJBs).
5. In our client's trade finance environment where event-based and remote business process execution are increasingly important, Kernl brings great benefits through its event-scheduling and messaging components. Kernl is a proven framework that is capable of facilitating hub and spoke processing when combined with a workflow product such as MQ Workflow.

7. In Summary, Why Choose Kernl?

1. Without a proven J2EE architecture solution, development teams face higher cost projects, project delays, inconsistent structure, little re-use and difficulty in maintaining applications. Kernl provides you with a flexible service-oriented, scalable and proven solution for your enterprise.
2. Kernl is built upon proven design patterns that help teach developers best practices for many complex J2EE problems.
3. Kernl reduces architecture design issues from your project's critical path. Architecture design decisions can occupy weeks of your architect's time – meanwhile, your development team is sitting idle waiting for the architecture to be completed.
4. With Kernl, the architects focus on issues critical to the business, and the development team can begin breaking down the requirements into design, and coding can be done immediately once the components and service beans are defined.