Klocwork

The Proven Leader. Software Security. Software Quality.

How to Minimize the Risks of Launching Modernization Projects

Klocwork Inc.
35 Corporate Drive
Burlington, MA 01803
1.866.556.2967
www.klocwork.com

Agenda

- Business drivers
- Modernization drivers
- The technical costs and risks
- The business costs and risks
- A method for de-risking and locking-in benefits
- Summary of benefits

A Successful Business needs to expand

Over time, the software base turns into a key strategic business asset

Business Drivers

- **Expanding market share**
- **Entering new markets**
- **Launch of Next generation products**
- **Competitive pressures**

The asset is leveraged to increase productivity and speed to market. It's a goose that lays golden eggs!

Klocwork
The Proven Leader. Software Security. Software Quality.

Expansion Requires Adaptation in Process

In the continued efforts to address markets and improve the bottom line, things begin to change …

- **Some software development is out-sourced**
- **Maintenance costs grow and 3rd line support teams are formed**
- **Development teams are larger and become distributed**
- **Platform variants are customized and put in separate streams**
- **Security becomes an important issue**

Klocwork
The Proven Leader. Software Security. Software Quality.

Modernization Drivers

Modernizing the existing software structure becomes both a need and an opportunity…

The need:

- ***productivity improvements***
- ***escaped defects to customer are persistent***
- ***new standards need to be supported***
- ***continual platform upgrades***
- ***increase software agility***

The opportunity:

- ***variants for new markets***
- ***new features for existing markets***
- ***capability extensions to out-perform competitors***

Klocwork
The Proven Leader. Software Security. Software Quality.

Architecture Woes!

But this is a fine time to find out that your software structure is "out of date and in a mess"…

- **The boundaries between components are blurred and even totally obscured**
- **APIs have been avoided or mis-used**
- **The interactions between components are complex and cyclically clustered**
- **No understanding of which components interact with others**
- **Features have been added that were not planned as part of the original architecture – they have been bolted on**
- **Functions have been cloned and renamed and are virtually untraceable**

The modernization that you need to undertake can suddenly become very risky and costly!

Klocwork
The Proven Leader. Software Security. Software Quality.

Moving Target and other Priorities!

It sounds pretty daunting, but this is a career opportunity to become a hero.

But even more trouble lies ahead…

- **The documentation is not up to date**
- **There are maintenance updates to deliver to improve quality**
- **New, inexperienced developers are on board with in-sufficient training while your SMEs are career developing on other projects**
- **Resources constantly pulled away for fire fighting**
- **The code is growing at an alarming rate**

Budgets and Executives…

On top of all this, there is a budget to fight for and executives to convince…

- **There are direct costs to consider:**
  - **Deployment of tools**
  - **Training to understanding your software**
  - **Addition of resources**
  - **On going maintenance of knowledge and architecture**
- **And opportunity costs:**
  - **Impacts to scheduled delivery of new functionality**
- **There is convincing to do:**
  - **Skepticism of need and ability to deliver**
  - **Fear of touching code**
  - **$$$ hard to free up**
  - **Lots of churn in lots of areas**

Klocwork
The Proven Leader. Software Security. Software Quality.

How do you start?

With the odds stacked against you, why would you "bet the farm" by performing your architecture driven modernization in one massive high risk, high cost venture?

A different approach is required…

…if you want to be successful!

Klocwork

The Proven Leader. Software Security. Software Quality.

A New Approach to Modernization

Incremental and "In Process"…

Architecture Driven Modernization should be undertaken using a controlled, incremental method that is completely integrated into the product development process.

AND…

Klocwork
The Proven Leader. Software Security. Software Quality.

Fully Managed as a Software Program

Managed with metrics

Architecture driven modernization activities should be managed by automatically mining objective, repeatable "Critical to Quality" indicators and metrics

"***Only in software do people cling to the illusion that it's OK to come up with estimates of the future, even though you've never measured anything in the past.***"
….Tom deMarco

Klocwork
The Proven Leader. Software Security. Software Quality.

Indicators

Critical to Quality indicators – those that provide measurable, repeatable, objective indicators reflecting quality, productivity and risk

Size

Defect Density

Risk

Clones

Churn

Structure Dependencies

Security Defects

Complexity

Klocwork

The Proven Leader. Software Security. Software Quality.

Incremental and In Process ADM: The Steps

1. Establish a Baseline
2. Start a Defect Reduction program for quick wins
3. Excavate architecture to surface anomalies
4. Establish model controls to prevent further erosion.
5. Select which modernization improvements will be done within this development cycle.
6. Track and manage (build over build) “Critical to Quality Indicators”
7. Manage and deliver incremental ADM completely within the product release cycle.

Step 1: Establish Baseline

- Chart out the trending for the “Critical to Quality Indicators” for the last 5 product releases
  - Has quality been improving or deteriorating?
- Use past values to establish realistic targets to achieve in future releases
- Publish current baselines and trends to set expectations and create awareness

Step 2: Start a Defect Reduction Program

- A defect reduction program will generate high initial value that augments the modernization values that will be achieved over time
- Focus on strategic defects
  - If memory leaks are a current sore spot with customers, then fix those first

Step 3: Excavate Architecture to Surface Anomalies

- **Excavate by aggregating software into components or layers of components. This creates the architectural model**
- **This has the effect of surfacing the key anomalies – the ones that couple together major components that should not be coupled**
- **Re-factor the model to remove the anomalies and capture the code changes required to implement**

Step 4: Prevent Erosion through Model Control

- **The results of the excavation produced an architectural model.**
- **Examine the model to determine which major components should have “uses” relationships with other components.**
- **When a “uses” relationship should NOT exist From component A to component B, set up an architectural rule that states that this relationship is NOT allowed.**
- **Detect and prevent rule violations by monitoring all updates/changes to the software build.**

Step 5: Prioritize Modernization Activities

- The results of the excavation produced an architectural model. Re-factoring shows what needs to be changed
- Assess changes for “Value”:
  - benefit to customer
  - extended life of code base
  - ease and velocity of feature development
  - reduced rate of defect insertion due to reduced complexity
- Trade of value and risk and create a release over release rollout plan (eg. Derisk by changing areas that are already changing for feature development)

Step 6: Track and Manage “Critical to Quality Indicators”

- **Measure the progress of changes through-out the development cycle to ensure program is tracking to achieve the predicted improvements**
- **Report on progress with objective metrics to help de-mystify architectural deliverables**
- **Provides excellent risk mitigation and cost control**

Step 7: Manage and Deliver within Existing Product Release Cycle

- Treat the ADM piece as a trackable feature
- Gets the same visibility as feature development
- Becomes part of the improvement culture
- Ensures new feature development doesn't erode the architecture

Benefits of Incremental and In Process ADM

- New features do not erode modernized architecture
- Architecture knowledge is always available to all developers and can be changed in a controlled manner to support new components
- Up to date architecture means that architect extensions required for new features are readily understood and can be executed in a controlled manner. Effort, ROI and risks are quantifiable
- Project management is greatly improved with the ability to measure and manage software system parameters

Klocwork

The Proven Leader. Software Security. Software Quality.

Thank you

nrajala@klocwork.com