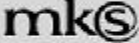


 <pre>public Sandbox getClosestSandbox(String fileName) { sifDirectoryServices sids = 5ifDirectoryServices();String absolutePath = null;File abs = null; if (fileNar</pre>	
	<b>Not for the fainthearted ..</b>
<pre>Absolute(getContext().new File(fileName));absolutePath = NativeFile.getCanonica atNativeDirectory(absolutePath,false);) else {absoluteDir = getContext().getCw</pre>	<p>Managing Configurations of Large, Complex Projects</p> <p>Linda Newsom-Ray and Tom Brett</p>


	
	<h2>Today's Agenda</h2> <ul style="list-style-type: none"><li>§ To discuss some of the issues and common problems involved in managing the configurations of large, complex projects</li><li>§ To share our experiences, with real examples, of how to overcome these problems</li></ul> <p>Build Better Software™</p>

mk<sup>®</sup> 

## Some typical complexities

- § Project scope
  - § business objectives far-reaching & business-critical
  - § technical objectives very ambitious / high-risk
- § Project environment
  - § development over a period of months, if not years
  - § requirements changing over time
  - § continuous pressure on time-to-market


Build Better Software™

mk<sup>®</sup> 

## Some of the challenges ..

- § Project management
  - § many people involved, many locations
  - § cultural and/or language differences
  - § multiple time-zones involved
  - § inter-dependent projects / sub-projects
- § Control
  - § poor communication paths
  - § processes not defined / rigorously enforced
  - § planning and tracking insufficient
  - § damage caused by 'brinkmanship'

Build Better Software™


mk<sup>®</sup> 

## Why these?

**Q:**  
Surely we're talking CM here, not general project management?

**A:**  
Poor project management **can** wreck a project - but poor CM within a complex project **WILL** wreck it

Build Better Software™


mk<sup>®</sup> 

## Damage caused by poor CM

Without strong Configuration Management control over each and every change within a project, there can never be a successful project

Any resulting application(s) will be unmaintainable

Build Better Software™


mkS 

## Issues caused by poor CM .. 1

No control over status/location of objects:

- § objects 'disappear' between environments
- § testing staff are testing wrong versions
  - § bugs that were fixed suddenly reappear
  - § wrong versions are released into production
- § uncontrolled objects included in build
- § objects in production for which there is no source code!

Build Better Software™

mkS 

## Issues caused by poor CM .. 2

Loss of project control

- § late changes introduced without formal control cause:
  - § confusion
  - § re-work
  - § over-runs
- § scope-creep may mean that project never finishes


Build Better Software™

mk<sup>®</sup> 

## IT-dependent organisations need to ..

- n co-ordinate activities of (many) large teams of developers and testers
- n manage parallel development
- n manage assembly of software releases from defined functional changes
- n deploy applications into multiple environments


Build Better Software™

mk<sup>®</sup> 

## So they must provide ..

- n infrastructure for communicating status of:
  - n project plans
  - n Configuration Management plans
  - n change requests
  - n other critical work-products
- n facilities for sharing code across:
  - n multiple development departments
  - n multiple test sites
  - n usually across multiple platforms
- n strategies for geographic collaboration and dispersal

Build Better Software™


mk<sup>©</sup> 

## What is important in 'grown-up' CM?

A quick reminder of the Capability Maturity Model<sup>(R)</sup> goals for SCM

- 1 Configuration Management activities are **planned**
- 2 Software work products are **identified, controlled and available**
- 3 **Changes** to identified work products are **controlled**
- 4 **Affected groups** are informed of the **status and content** of software baselines


Build Better Software™

mk<sup>©</sup> 

## Overcoming the issues

- n The key to managing large, complex projects is to start with a process-based solution
  - n macro processes - organisation / project
  - n micro processes - project / developer
- n The CM activities within any complex project must **SUPPORT** the process, not impede it
  - n CM planning
  - n defined organisational responsibilities (e.g. SCCB)
  - n visibility of completed baselines (e.g. high-level design)
  - n absolute control over what is moved to where, and why and when

Build Better Software™


mk<sup>©</sup> 

## Macro processes - planning & control

- n **Provide visibility and control**
  - n plan functional content of release
  - n plan quality processes
    - n e.g. Reviews; SCM functional and physical audits
  - n define authorisation levels at each stage
  - n define tracking points, for progress control
- n **Understand the scope**
  - n change clearly defined - in-scope and out-of-scope
  - n scope & requirements understood by people implementing change
  - n impact of 'changes to the change' understood *before* anyone acts on them!

NB Provide supporting tools, to ensure compliance

Build Better Software™

mk<sup>©</sup> 

## Micro processes - within the projects

**Record all code-level changes**

- n total-lifecycle traceability of code-level changes to the reason for change
- n maintain compatibility between different changes
- n prepare for releases
- n apply changes to different environments or streams of development
- n track progress against functional requirements
- n **make quality happen!**

**NB MUST provide robust supporting tools**

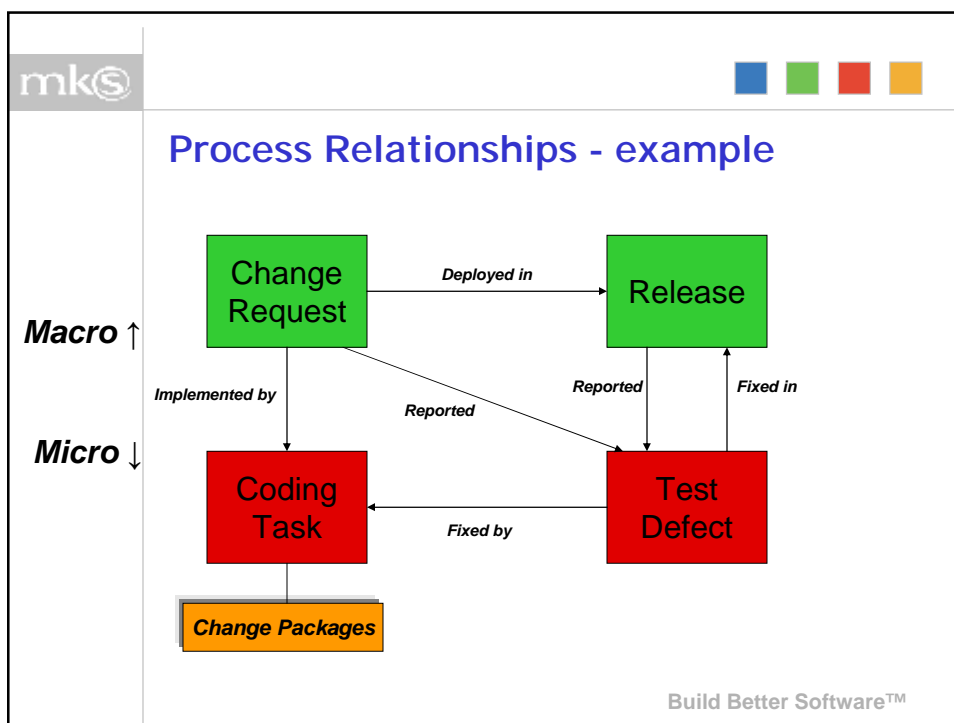
Build Better Software™

mkS


## Macro/Micro process split

- n **Macro Processes**
  - n Must contain authorisations and controls
  - n Often (necessarily) heavyweight
- n **Micro Processes**
  - n Quick, simple, lightweight
  - n Like "tight loop" code!
- n "One to Many" relationship
- n Frequent source of confusion!

Build Better Software™






mk<sup>®</sup> 

## Macro / Micro Process – getting it right!

- n CRITICAL to success of an SCM implementation!
- n No “one size fits all”
- n “Best Practice” = what suits your organisation
- n Requires in-depth planning


Build Better Software™

mk<sup>®</sup> 

## Managing parallel development

- n **What is parallel development?**
  - n ... development which diverges from a common starting point
  - n Also implies the possible need to converge again
- n **Why?**
  - n Post-release maintenance
  - n Experimental or custom development
  - n Different product variants from same code base
- n **Macro/Micro**
  - n Macro: long-term parallel projects
  - n Micro: branch at member level for concurrent changes


Build Better Software™

mkS 

## Branch per Change

- n **Principles**
  - n All changes take place on branches
  - n Functional segregation of branches
  - n Only QA'd code is merged to the "trunk"

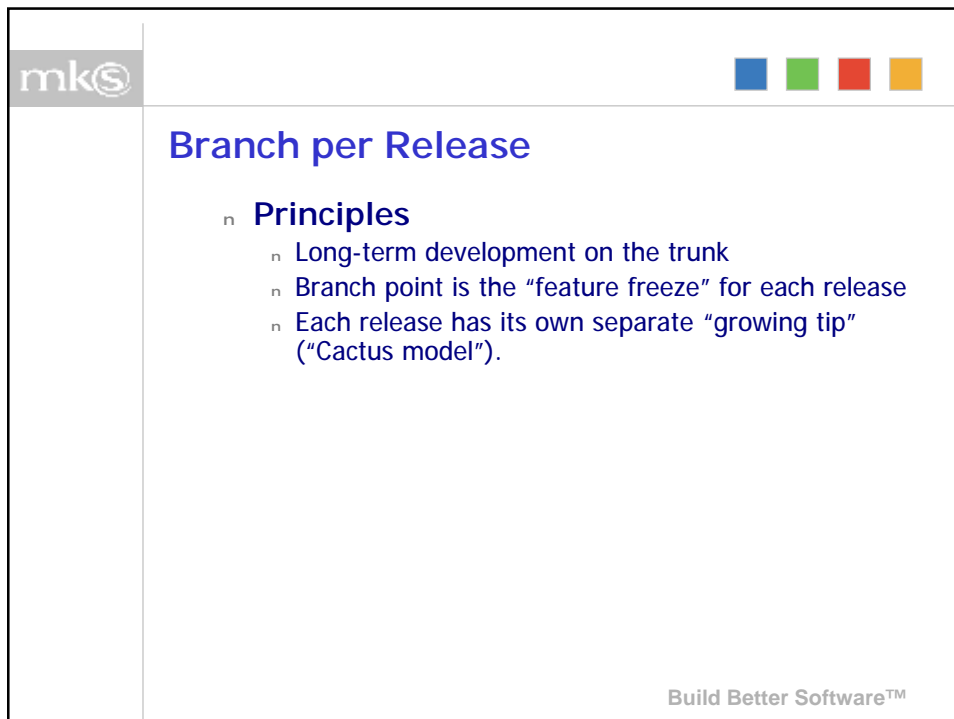
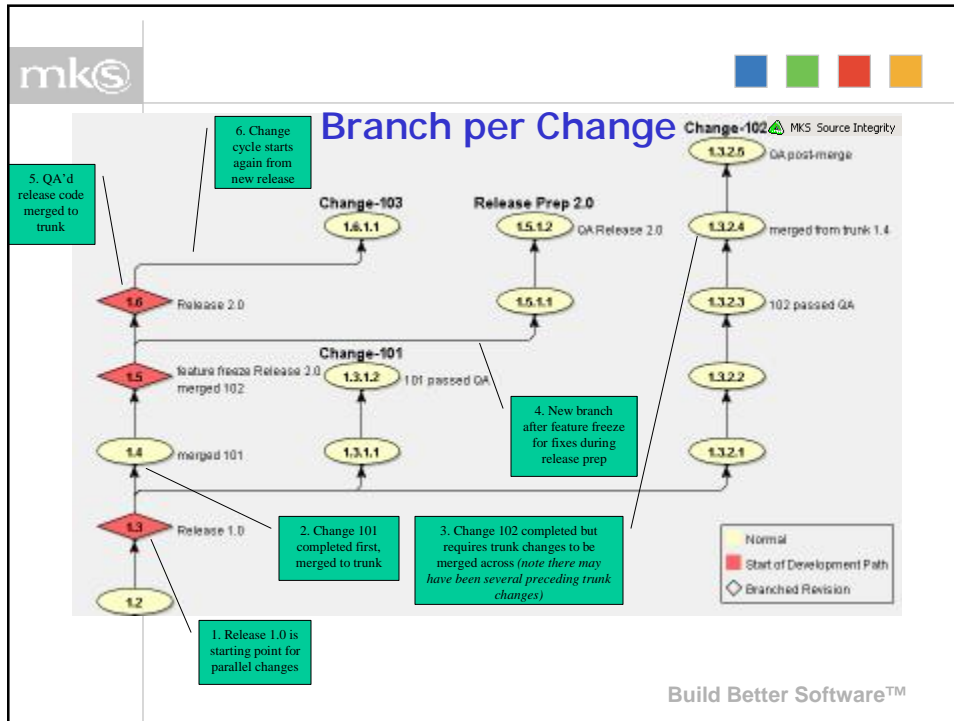
Build Better Software™

mkS 

## Branch per Change

- n **Advantages**
  - n Compact trunk (readability)
  - n All releases on trunk, easy to follow release history
- n **Disadvantages**
  - n Requires large & complex merges
  - n May require duplication of QA (before & after merge to trunk)
  - n Imposes cumbersome process, even for small changes
    - Concurrent changes introduce additional merge overhead
    - Risk of overwriting previous changes

Build Better Software™



mkS

Branch per Release


- n **Advantages**
  - n Relatively small and simple merges
  - n Post-release fixes do not require a merge.
  - n Simple QA
- n **Disadvantages**
  - n Releases spread across different branches
  - n Fixes may need to be applied to more than one branch
  - n Changes for different features may be interleaved on same branch

Build Better Software™

mkS

Branch per Release


Build Better Software™

mk<sup>®</sup> 

## Branch per Environment

- n **Principles**
  - n Separate branch for each environment to which code is deployed (development, test, production, standby...)
  - n Tip of each branch is the current configuration
  - n Deployment of code achieved by applying change to the target branch

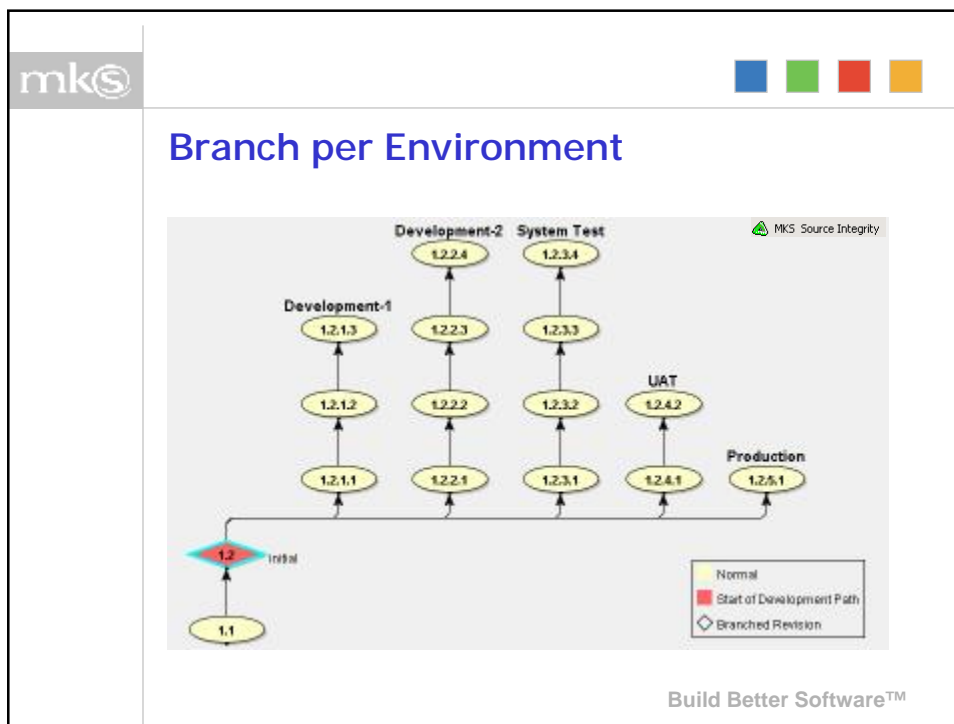
Build Better Software™

mk<sup>®</sup> 

## Branch per Environment

- n **Advantages**
  - n Straightforward deployment
  - n Highly visible history for each environment
  - n Mostly avoids need to merge code
- n **Disadvantages**
  - n High maintenance
  - n Changes for different features may be interleaved on same branch
  - n No segregation between "releases"
  - n Best suited to incremental, "in-house" development

Build Better Software™



### "Patterns" in SCM

- n Patterns: "kinds of" engineering structures to solve classes of problems.
- n Engineering analogy:
  - PROBLEM: River crossing
  - PATTERNS: Bridge; Ford; Ferry; Tunnel
- n Allows engineer to draw on body of experience

Berczuk & Appleton  
*Software Configuration Management Patterns*  
Addison-Wesley, 2002

Build Better Software™

mkS

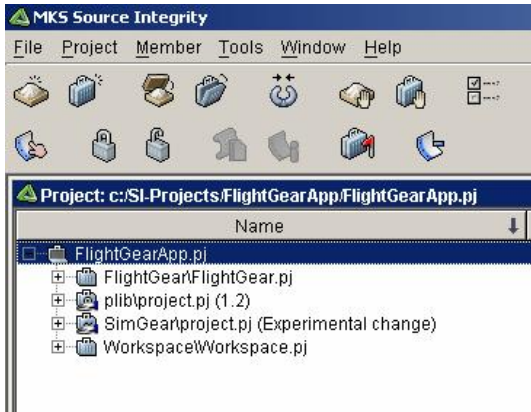
Application Structure - constraints

- n Source structure for build
- n Runtime directory/path structure
- n Shared or re-used code
- n Branching pattern(s) in use
- n Deployment pattern(s) in use

Build Better Software™

mkS


Re-usable / configurable components



The screenshot shows the MKS Source Integrity application window. The title bar reads 'MKS Source Integrity'. The menu bar includes 'File', 'Project', 'Member', 'Tools', 'Window', and 'Help'. Below the menu bar is a toolbar with various icons for project management. The main area displays a project browser for 'Project: c:\SI-Projects\FlightGear App\FlightGear App.pj'. The tree structure is as follows:

- FlightGearApp.pj
  - FlightGear\FlightGear.pj
  - plib\project.pj (1.2)
  - SimGear\project.pj (Experimental change)
  - Workspace\Workspace.pj


Build Better Software™

mk<sup>®</sup> 

## Software Release assembly

- n Macro meets Micro Process
  - n Macro Process defines the release / deployment cycle
  - n Micro Process defines the actual code-level changes
- n Deployment Requirements
  - n Full system build (CD / shrink-wrap)
  - n Patches
  - n In-house deployment
- n Choose release/deployment pattern to meet requirements


Build Better Software™


mk<sup>®</sup> 


## Deployment by System Build

- n CM "Checkpoint" defines full set of application source code
- n Deployable application built from source
- n Rework at any stage results in a new checkpoint
- n Best suited when runtime system is hard to change (e.g. embedded systems)

Build Better Software™

mk <sup>®</sup>	
	<h2 data-bbox="493 394 1052 432">Deployment by Runtime Code</h2> <ul data-bbox="545 478 1218 888" style="list-style-type: none"><li>n Similar to "System Build" EXCEPT that built (runtime) code is also versioned under CM</li><li>n Deployment based on Checkpoint of the runtime code</li><li>n Ensure correspondence between source code checkpoint &amp; runtime checkpoint</li><li>n Rework at any stage results in new checkpoints of both source &amp; runtime</li><li>n Best suited when runtime system may need to be refreshed quickly without build from source (e.g. in-house development)</li></ul> <p data-bbox="989 947 1227 972">Build Better Software™</p>


mk <sup>®</sup>	
	<h2 data-bbox="493 1222 971 1260">Deployment by Exception</h2> <ul data-bbox="545 1333 1179 1604" style="list-style-type: none"><li>n Deploy only the changed code</li><li>n Variation of "Deployment by Runtime"</li><li>n Create a checkpoint with only the changed items</li><li>n Best suited to in-house development</li><li>n Note special handling when items are removed as part of the change</li></ul> <p data-bbox="989 1776 1227 1801">Build Better Software™</p>

mk<sup>®</sup> 

## Cross-Platform operation

- n **Process** should be platform-independent
- n **Branching model** should support variant platform-dependent builds
- n **CM Tool** support
  - n Platform-independent repository
  - n Support for multi-platform file formats
  - n Portable UI

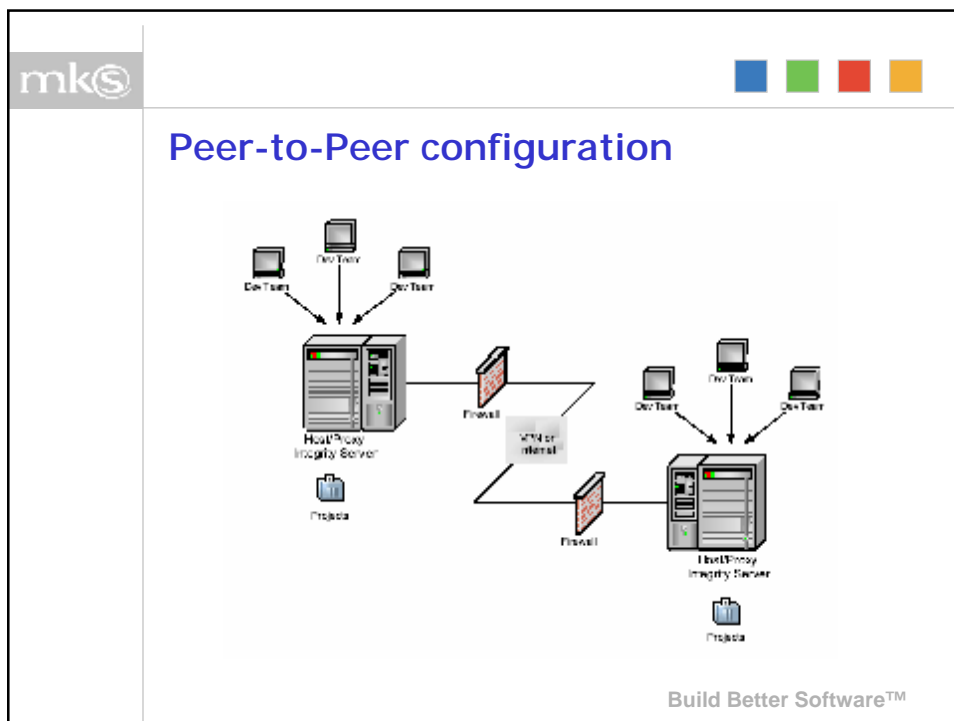
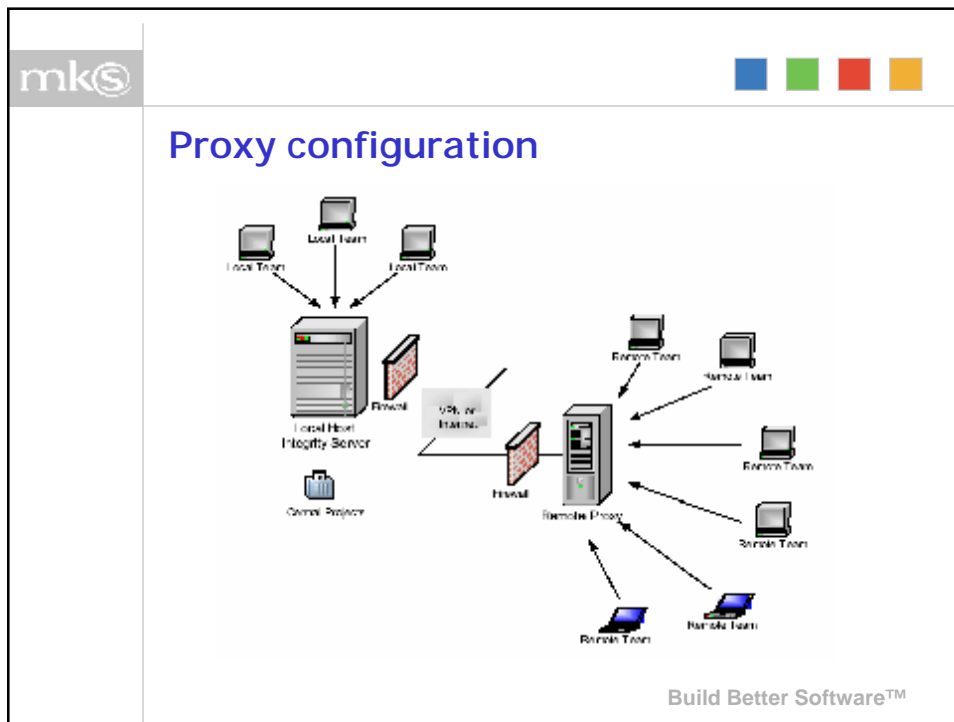
Build Better Software™


mk<sup>®</sup> 


## Geographic dispersal

- n **All sites** follow the same process
- n **Dispersal strategies**
  - n Central server
  - n Replication
  - n Proxy
  - n Peer-to-Peer
- n **Code integration**
  - n Standards conformance
  - n Tool-supported
- n **Access control**

Build Better Software™



mk <sup>®</sup>	
	<h2 data-bbox="493 394 753 436">To sum up .. 1</h2> <p data-bbox="545 487 1162 558">Managing configurations successfully within large, complex projects requires:</p> <ul data-bbox="594 621 1198 781" style="list-style-type: none"><li data-bbox="594 621 1198 718">n <b>processes</b> that are <i>well-defined, matched to the organisation's needs, and consistently applied</i> at the <b>macro</b> level and the <b>micro</b> level</li><li data-bbox="594 747 1075 781">n <b>structures</b> that support the processes</li></ul> <p data-bbox="987 949 1230 974">Build Better Software™</p>

mk <sup>®</sup>	
	<h2 data-bbox="493 1222 753 1264">To sum up .. 2</h2> <p data-bbox="555 1314 1211 1348">Managing configurations successfully requires:</p> <ul data-bbox="594 1398 1162 1684" style="list-style-type: none"><li data-bbox="594 1398 1162 1537">n <b>industry-strength tools</b> that provide developers, testers and management with <b>visibility and control</b> over the changes to the code, so that its status is known at all times</li><li data-bbox="594 1579 1162 1684">n the <b>organisational will</b> to keep 'banging the drum' to make sure that the standards are upheld</li></ul> <p data-bbox="987 1780 1230 1806">Build Better Software™</p>

