Source Code Control and Software Reuse

Presented by: Steven Hoenig

Business Unit Manager

Bloomy Controls Inc.

OverVIEW

- About Bloomy
- Challenges of Large Applications
- Source Code Control (SCC)
- Bloomy SCC Best Practices and Case Studies
- Software Reuse
- Demos
- Questions

***** Material borrowed from NI's presentation 'Best Practices for Software Development and Source Code Management'***

About Bloomy

Background

Bloomy Controls provides turnkey systems for automated test, data acquisition, and control

- Founded in 1992
- Facilities located in:
  - Windsor, CT
  - Marlborough, MA
  - Fair Lawn, NJ
- 45 Full-time, permanent employees
  - **27 Engineers**
  - 4 Project Managers
  - 4 Technicians
  - 10 executives, purchasing, sales, marketing
- Partnerships and temporary employment as needed to meet fluctuating demand

Accreditations

- Company
  - NI Select Alliance Partner (1998)
  - NI Certified Training Center (3 locations)
  - ISO9001:2008 Registered
  - ITAR registered
- Technical staff
  - Engineers:
    - NI Certified Architects (21)
    - NI Certified Developers (16)
    - NI Certified Professional Instructors (18)
  - Technicians
    - IPC-A-610, IPC-A-620, and J-STD-001D Certification

NATIONAL INSTRUMENTS Platinum Alliance Partner

ISO 9001:2008 CERTIFIED

NI Award:

Most Outstanding Technical Resources 2014, 2013

The most NI Certified engineers in the world!

BLOOMY

Software Development

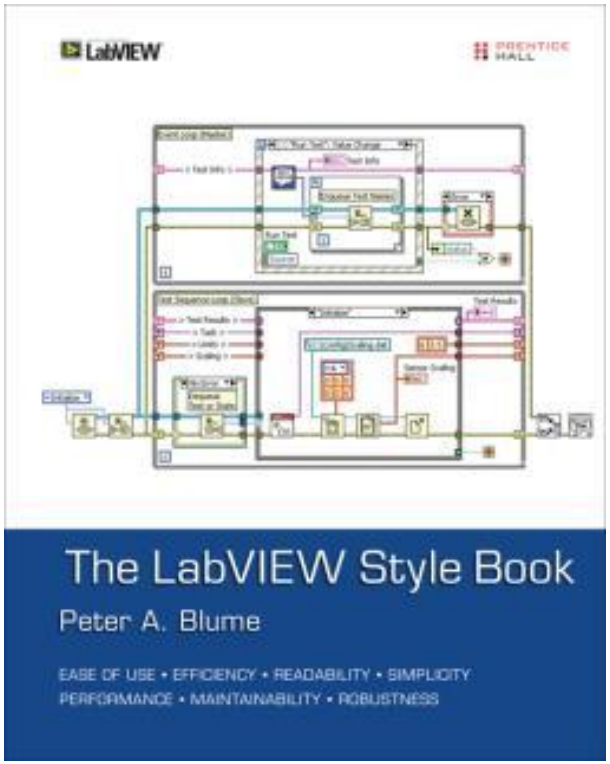
- NI Software Platforms
  - LabVIEW
  - TestStand
  - LabWindows/CVI
  - Real-time, FPGA
  - VeriStand
  - C#
- Data management / Database design (SQL)
- Data Analysis and reporting
- Training
  - LabVIEW, TestStand, LabWindows/CVI, DAQ
  - Customized on-site classes

BLOOMY

The LabVIEW Style Book

> The *LabVIEW Style Book* is the definitive guide to best practices in LabVIEW development.

Jeff Kodosky
Inventor of LabVIEW and Business and Technology Fellow
National Instruments



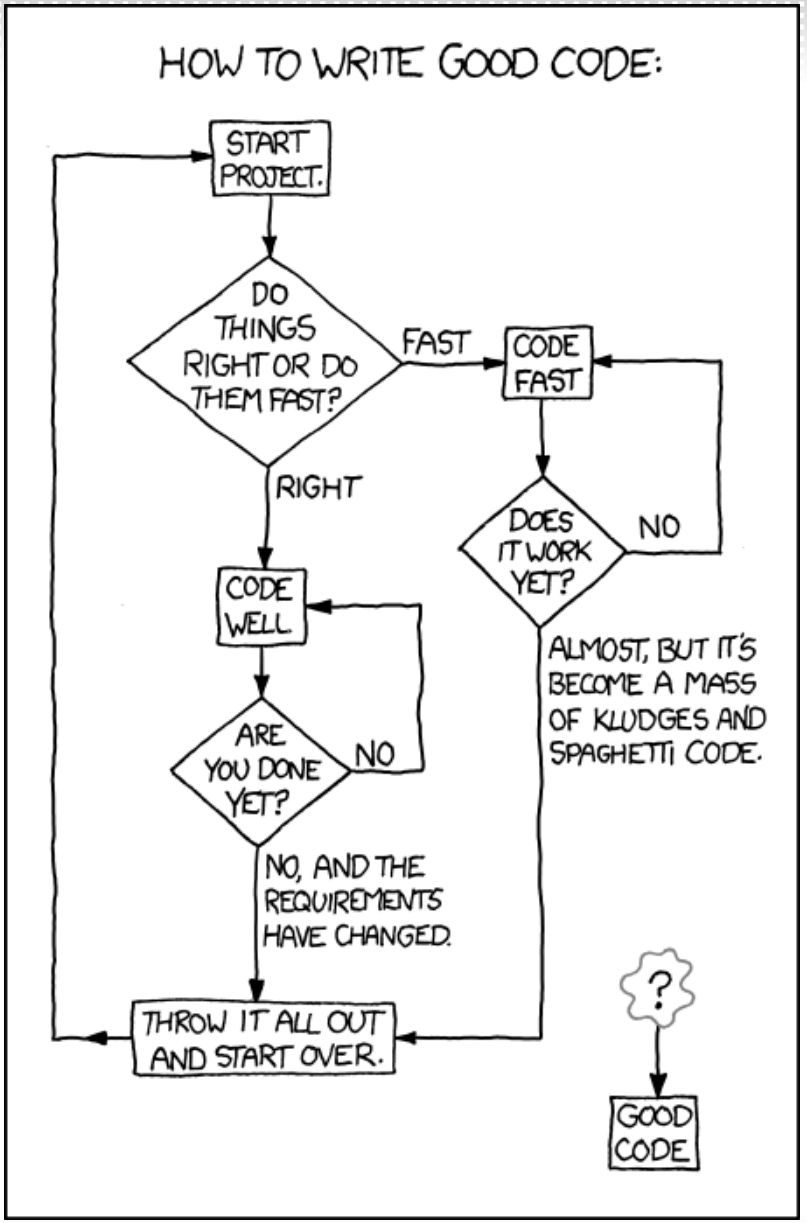
- Design patterns
- Data structures
- Error handling strategies
- Documentation
- Code reviews

Internal development standards

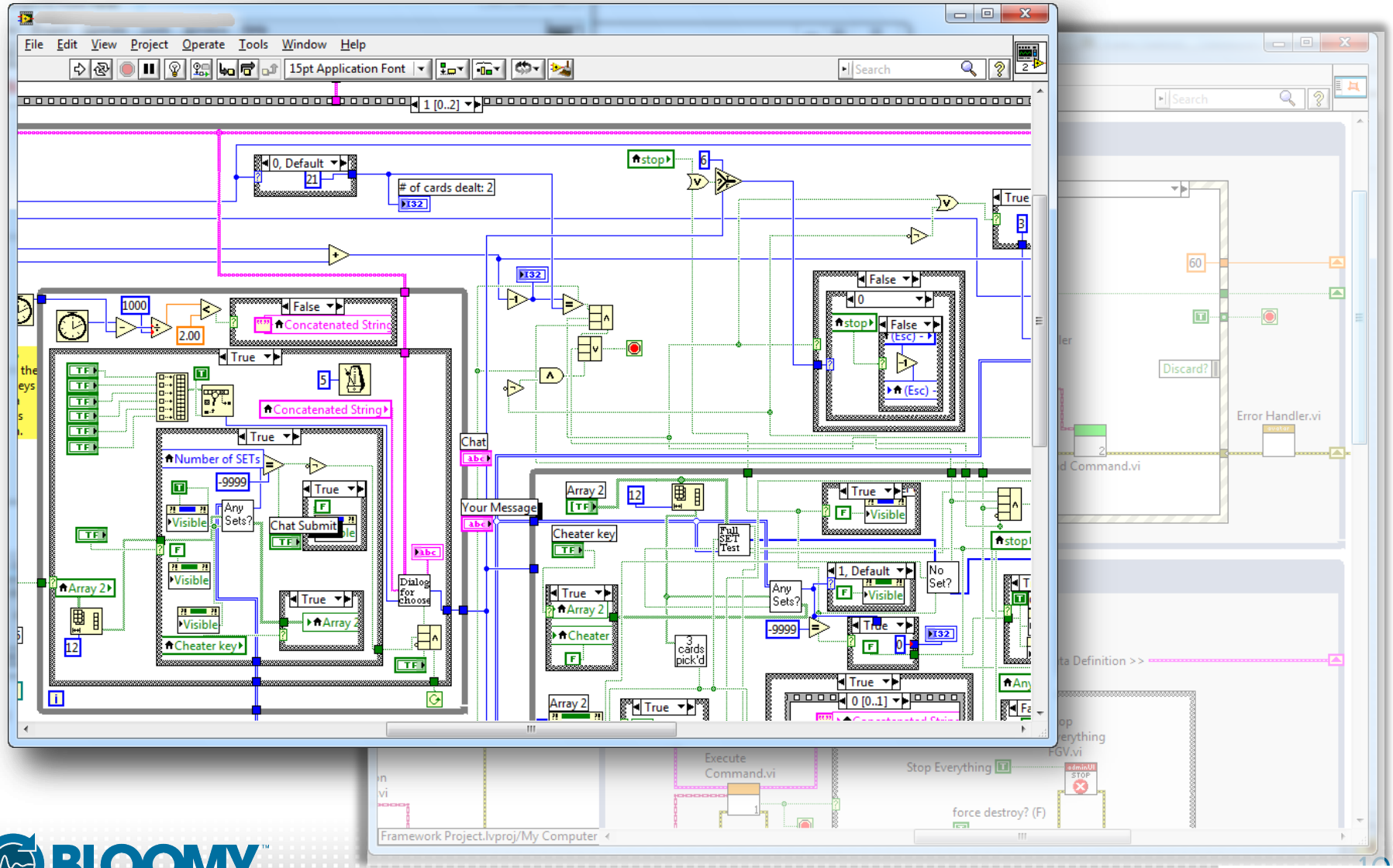
Why Do I Bring This All Up ?

- Givens:
  - 10's of software development projects active at any time
  - 1000's of LabVIEW files in a typical project
    - 100's of other standard project files (e.g. specs, docs, pm)
  - 3-7 members on typical project , all contributing to the project package
  - Developers spread between 3 geographic locations
  - Heavy reliance on leveraging internal IP (design patterns, software reuse) between developers
- Challenge:
  - How to efficiently develop software and execute projects in this type of environment

Challenges of Large Applications



Challenges of Large Apps



Software Engineering Debt

(just *some* of the most common LabVIEW development mistakes)

- ✓ **No source code control (or Project)**
- ✓ Flat file hierarchy
- ✓ 'Stop' isn't tested regularly
- ✓ Wait until the 'end' of a project to build an application
- ✓ Few specifications, documentation, or requirements
- ✓ No 'buddying' or code reviews
- ✓ Poor planning (lack of consideration for SMoRES)
- ✓ No test plans
- ✓ Poor error handling
- ✓ No consistent style
- ✓ Tight coupling or poor cohesion

NATIONAL INSTRUMENTS

BLOOMY™

Cost of a Software Defect

Development Phase	Cost Ratio
Requirements	1
Design	3-6x
Implementation	10x
Development Testing	15-40x
Acceptance Testing	30-70x
Post Release	40-1000x

Based on an analysis of 63 software development projects at companies including IBM, GTE, and TRW

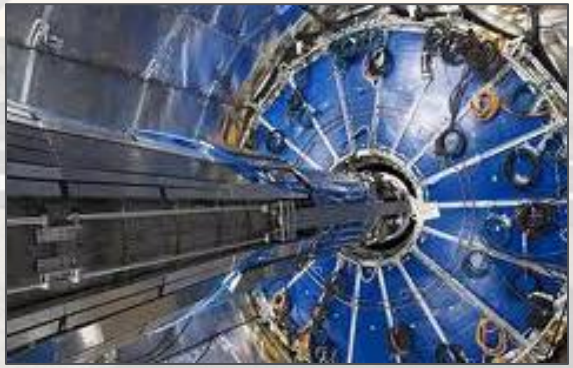
LabVIEW is used for Large Apps

High-Volume Production Test

Structural Health Monitoring

Medical Devices

Robotics and Mechatronics

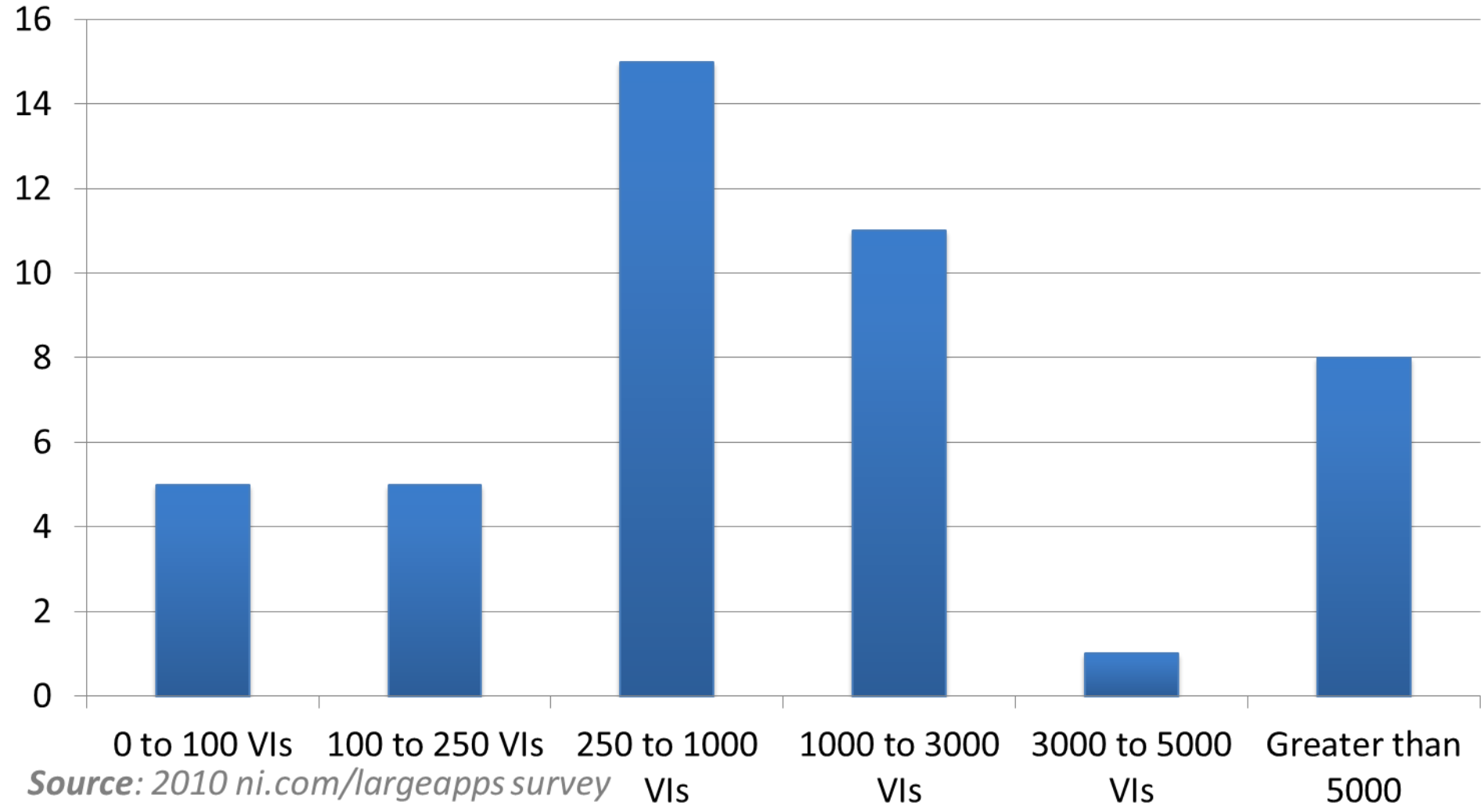


Large Physics Applications

Avionics Applications

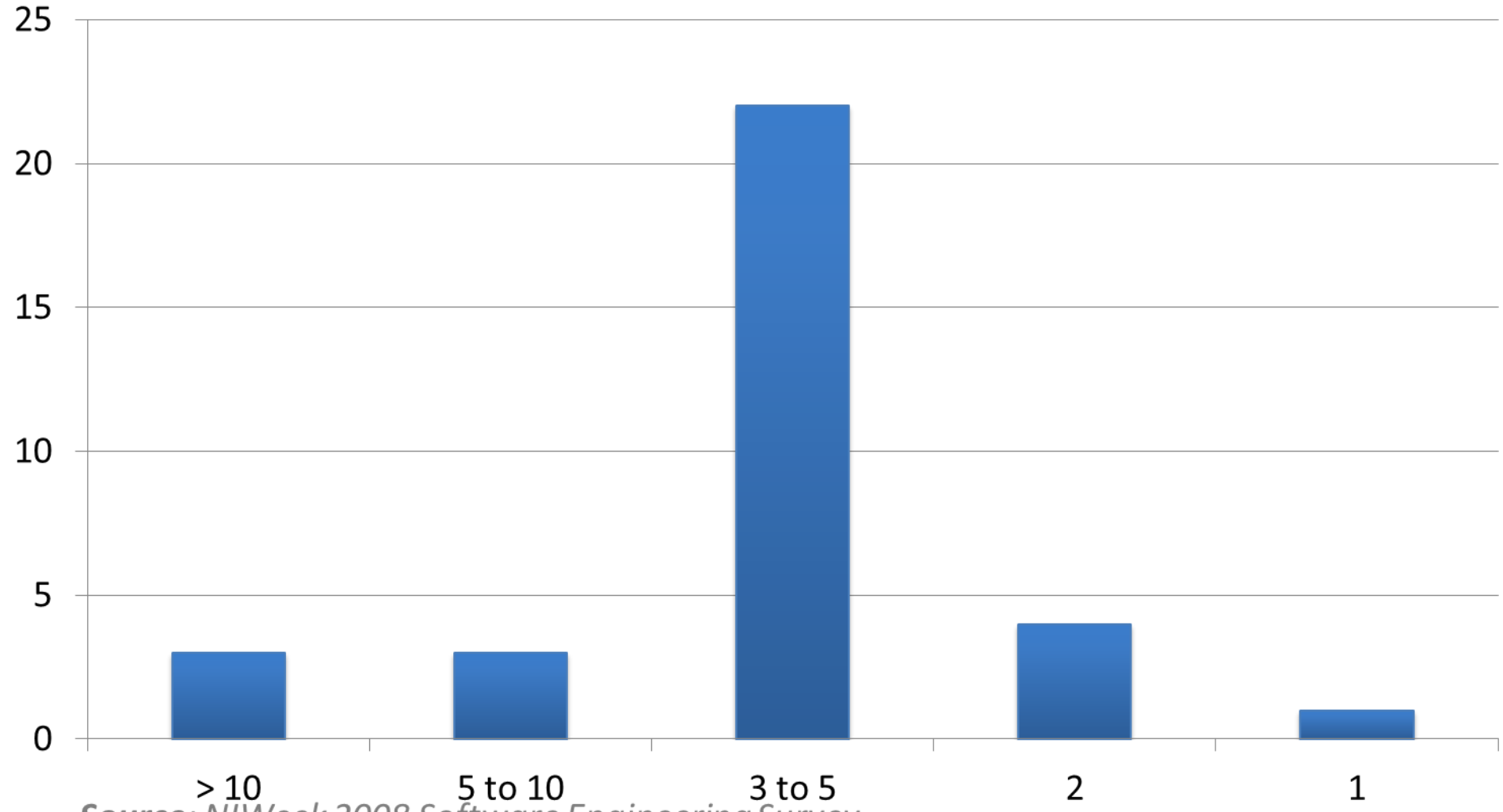
NATIONAL INSTRUMENTS

Size of LabVIEW Applications



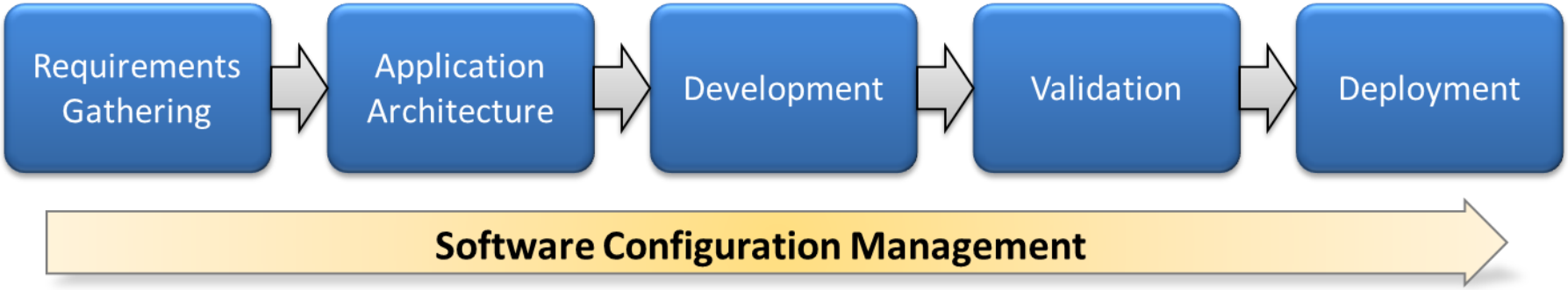
Source: 2010 ni.com/largeapps survey

Average Number of Developers Per Project



Source: NIWeek 2008 Software Engineering Survey

Software Configuration Management

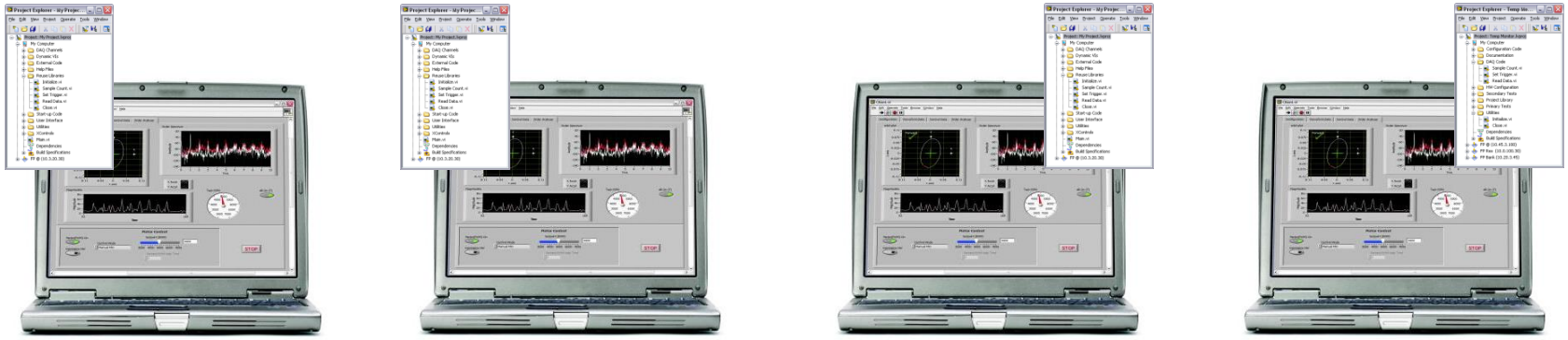


- Provides repository of code
- Helps manage source code and track changes
- Crucial for team-based development
- Important throughout development process

Configuration Management—Activities designed to monitor and control the evolution of a software product

Team Based Development

How can an entire team co-develop one application without stepping all over each other?



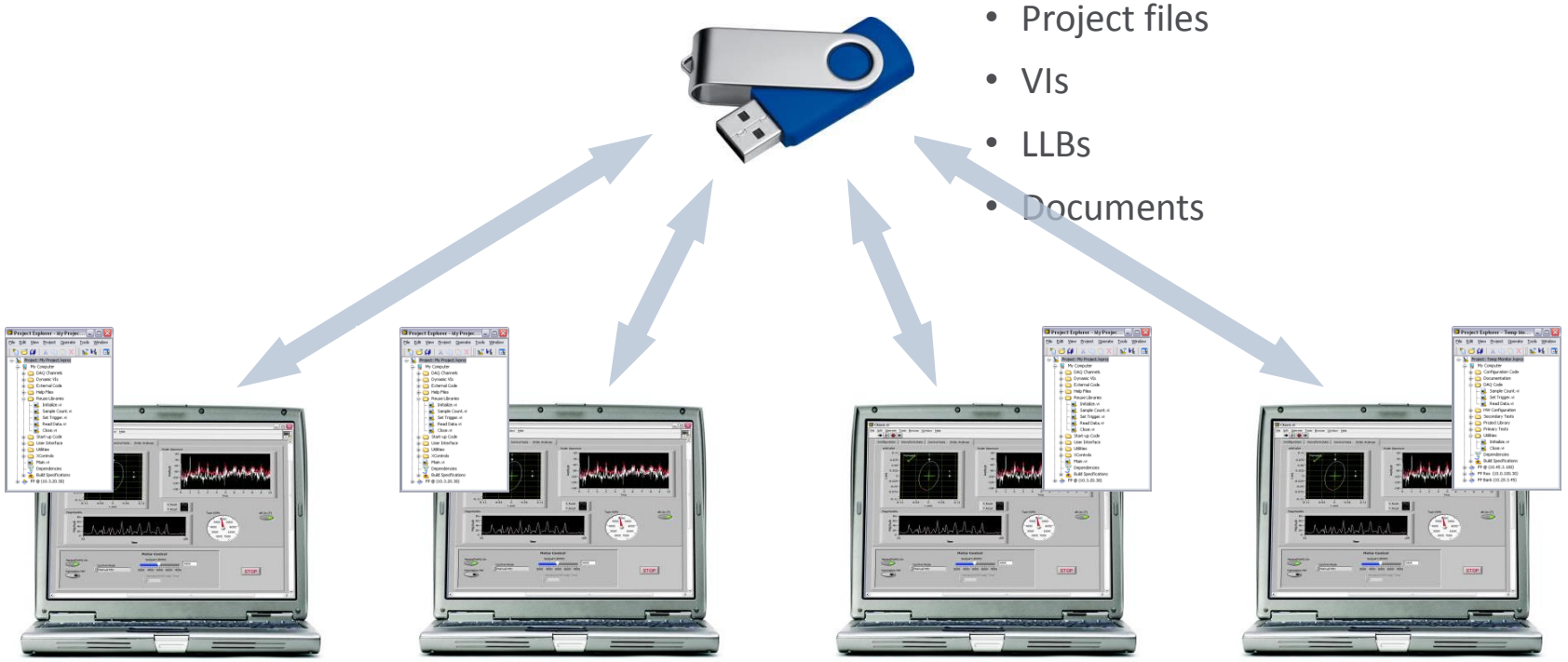
Each developer needs to share files with other developers

Team Based Development

Iterative Source Code Transfer

Transfer latest revisions of files

- Project files
- VIs
- LLBs
- Documents

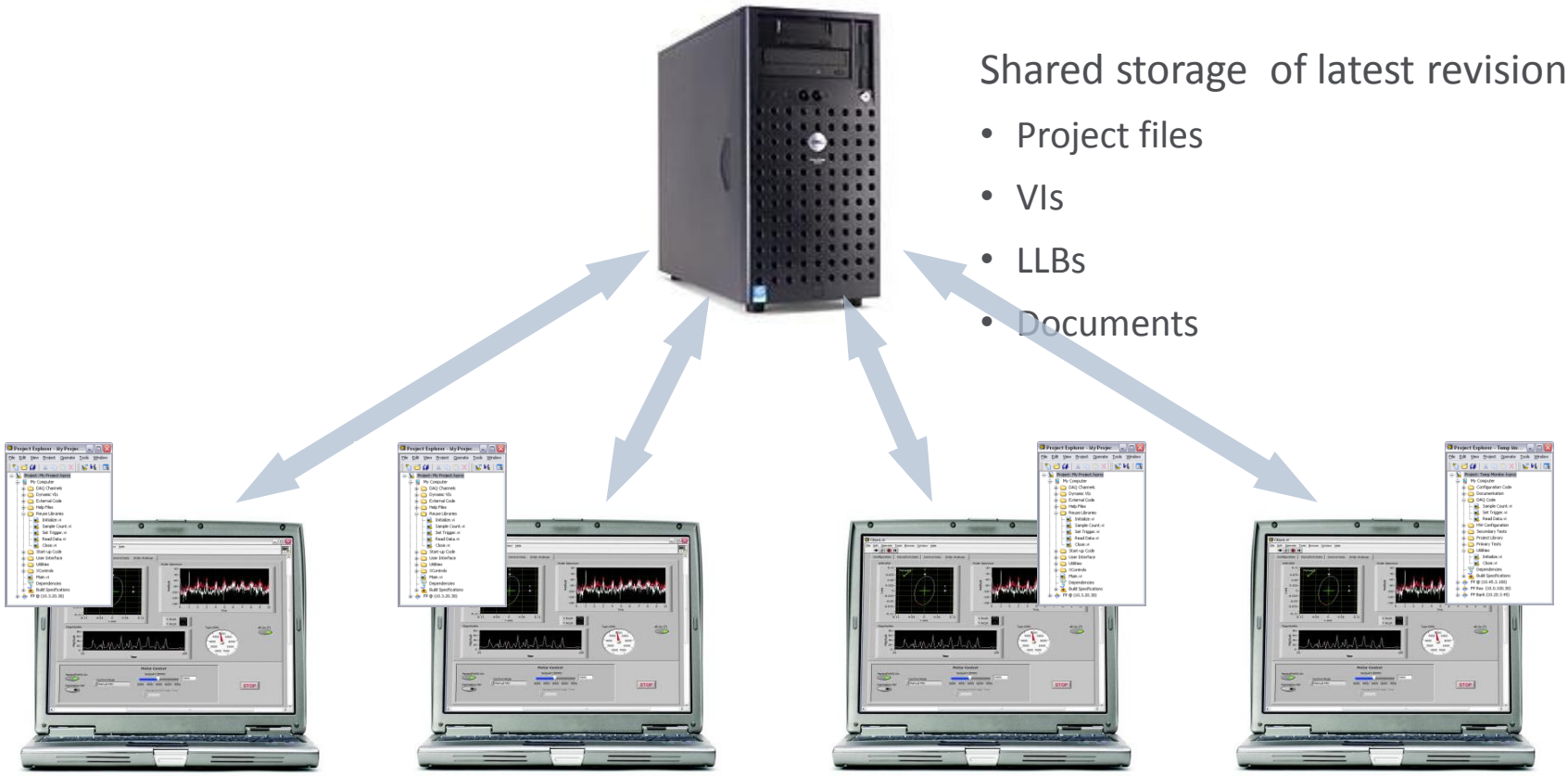


Each developer can share copies of files with others when ready

Who has the master copy?

Team Based Development

Centralized Source Code Storage



Shared storage of latest revisions

- Project files
- VIs
- LLBs
- Documents

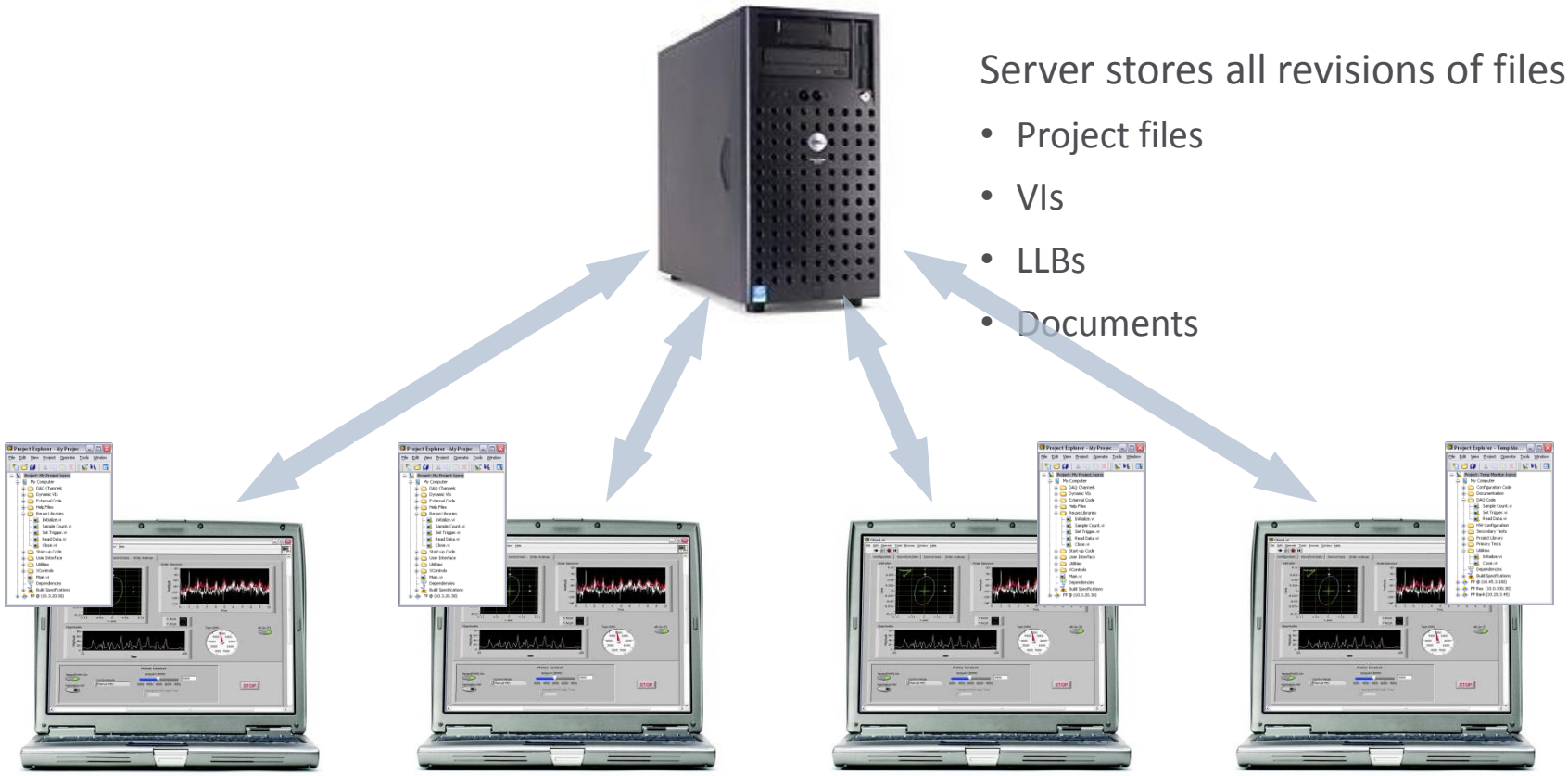
Each developer can check copies of files in and out as needed

Master Copy on Server – Who's putting everything together? What is the latest revision?

BLOOMY™

Team Based Development

Centralized Source Code Control



Server stores all revisions of files

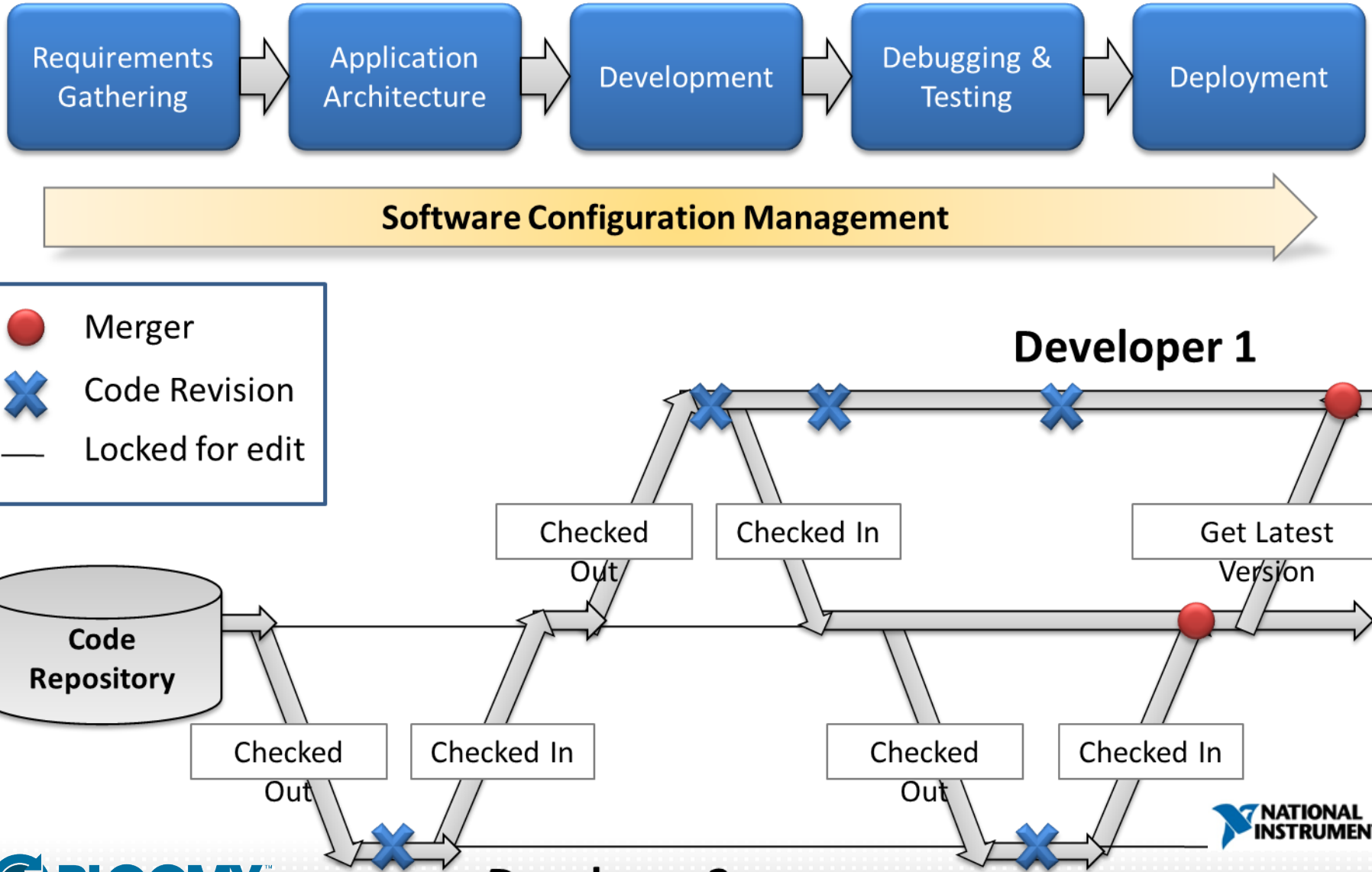
- Project files
- VIs
- LLBs
- Documents

Each developer can check copies of files in and out as needed

Source Code Control System manages storage, merges and updates / reversions

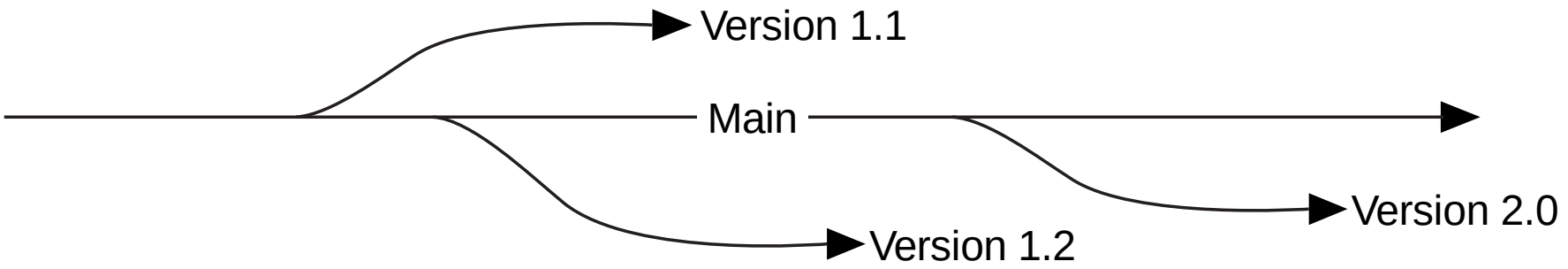
Source Code Control (SCC)

Source Code Control



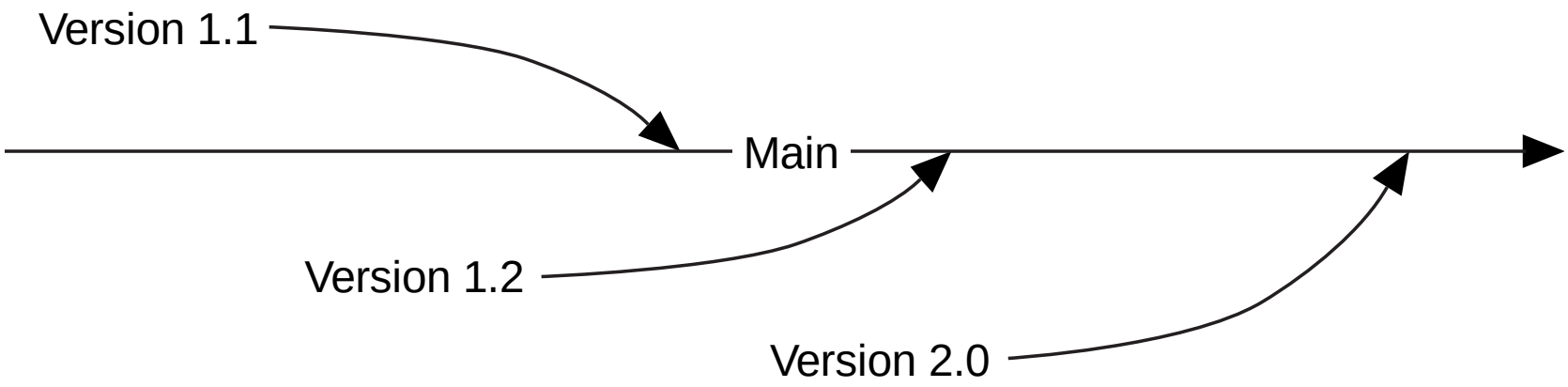
Branching Code

Branch—Split from the main development line to create a new version of the code



Merging Code

Merge—Integrate the development split into the main development line



SCC Options for Integration Within LabVIEW

Native LabVIEW Integration

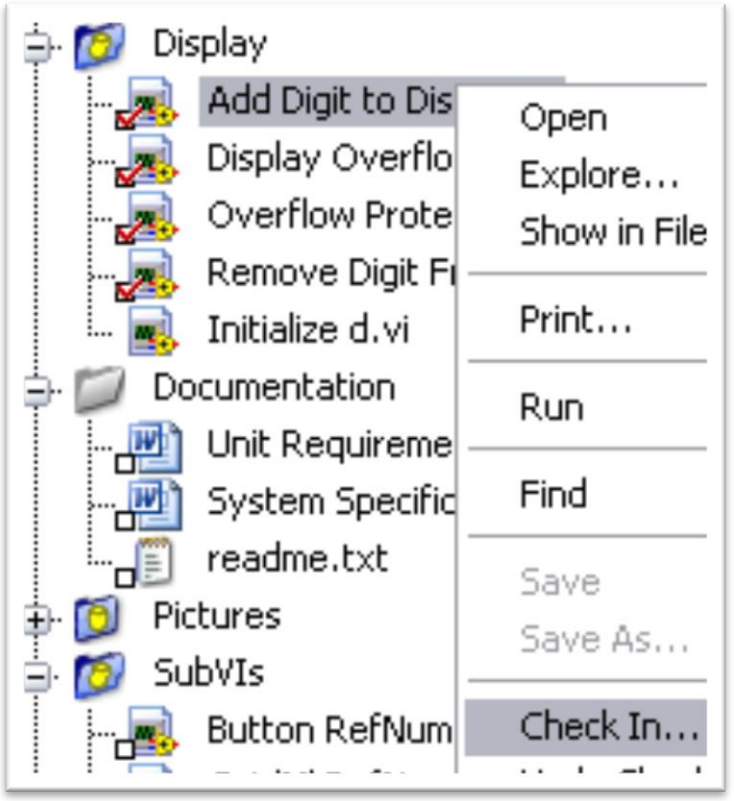
- Perforce

Integration Through Standard API

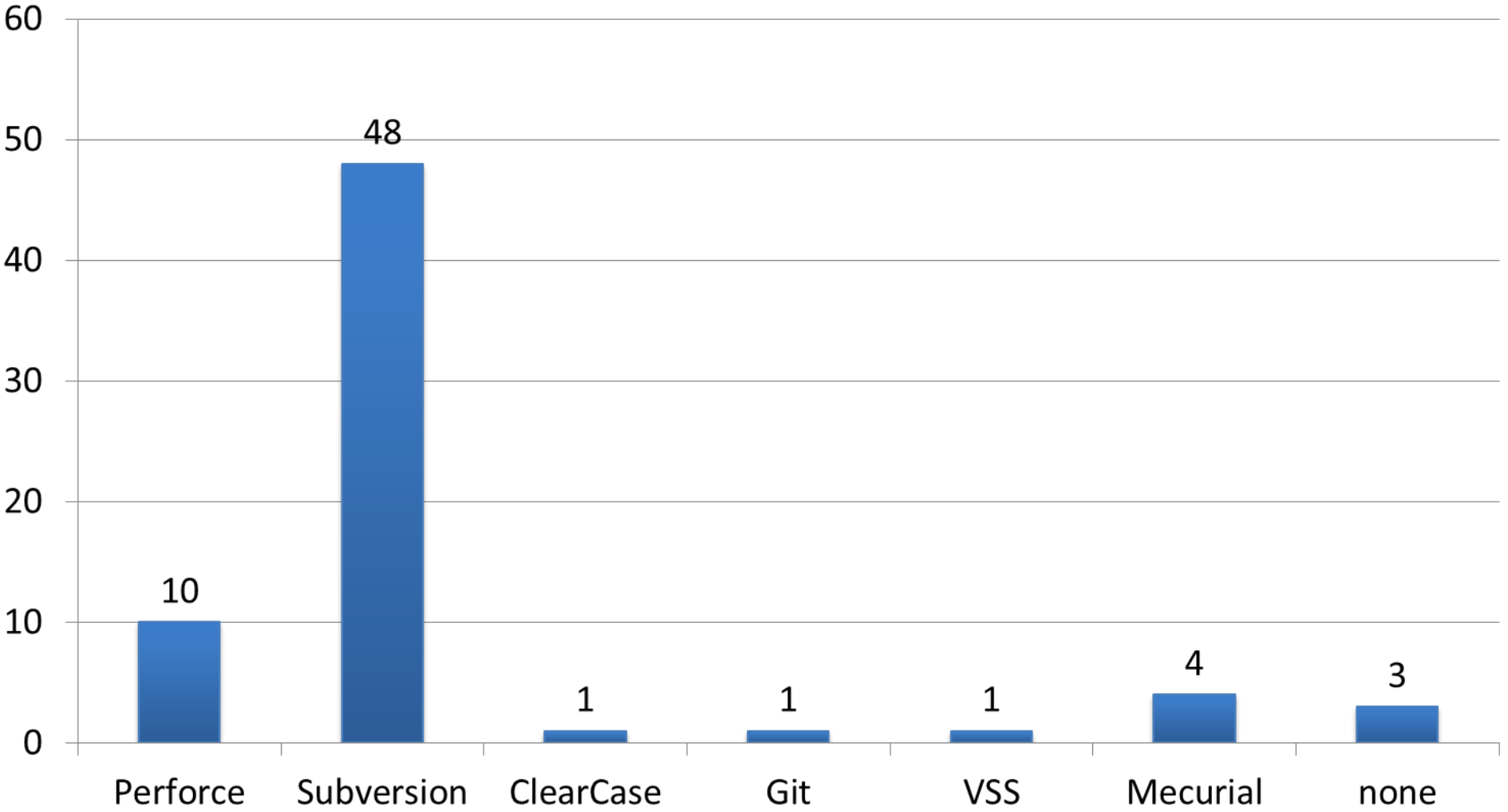
- Microsoft Visual SourceSafe
- Microsoft Team System
- Rational ClearCase
- PCVS (Serena) Version Manager
- MKS Source Integrity
- Seapine Surround SCM
- Borland StarTeam
- Telelogic Synergy
- ionForge Evolution

Support Through Additional Add-Ons

- Subversion
- Mecurial



SCC Options of LabVIEW Programmers



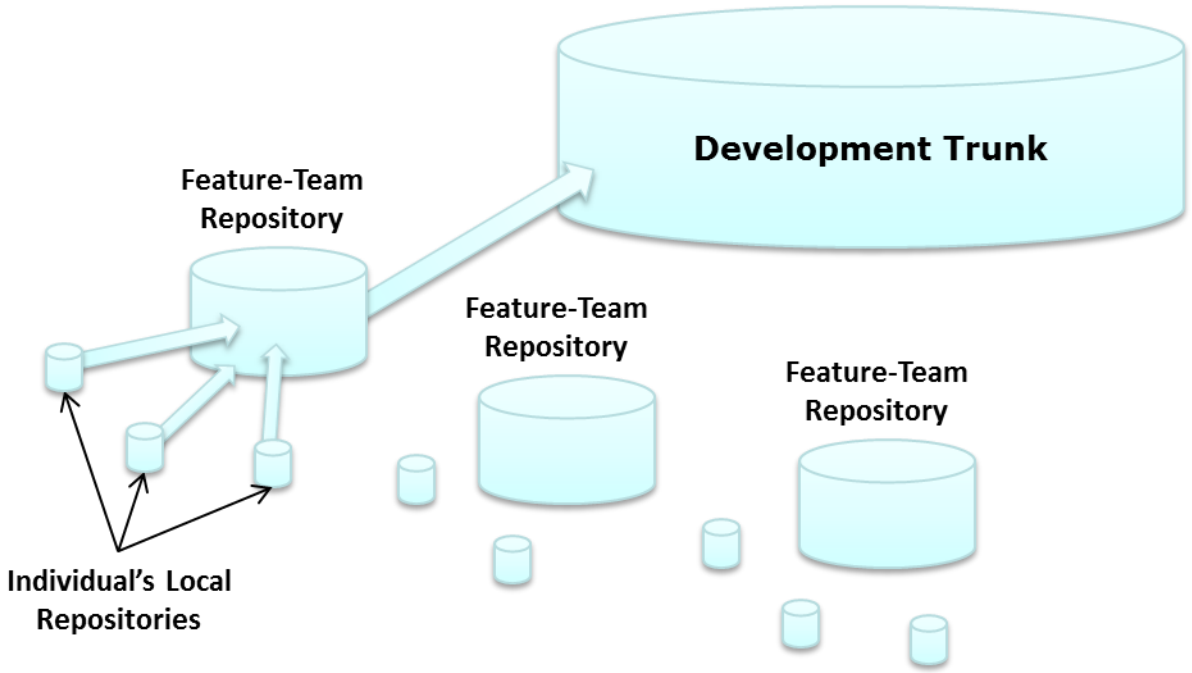
***Source**: 2010 ni.com/largeapps survey*

NATIONAL INSTRUMENTS

BLOOMY

NI Configuration Management

- Different trunk for each LabVIEW version
- Teams of 3 to 7 developers work in smaller repositories
- Individuals may have their own repositories
- New features and changes are regularly merged

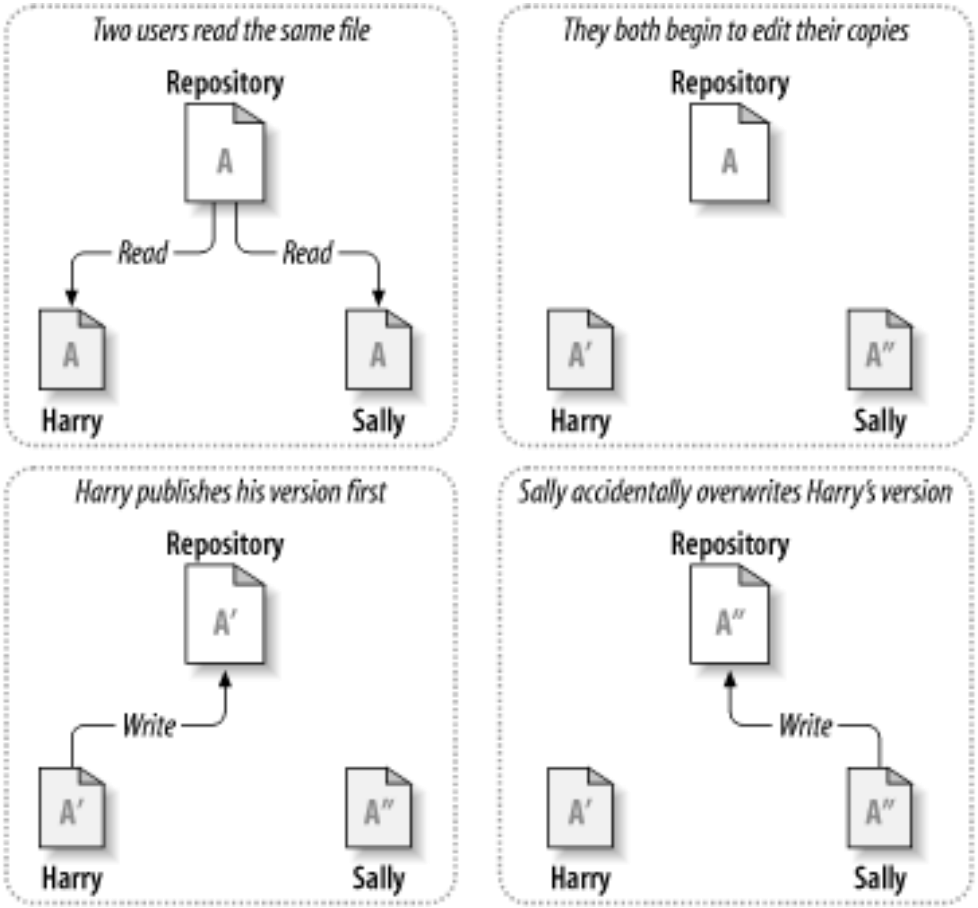


Best Practices /Case Studies

Why Source Control

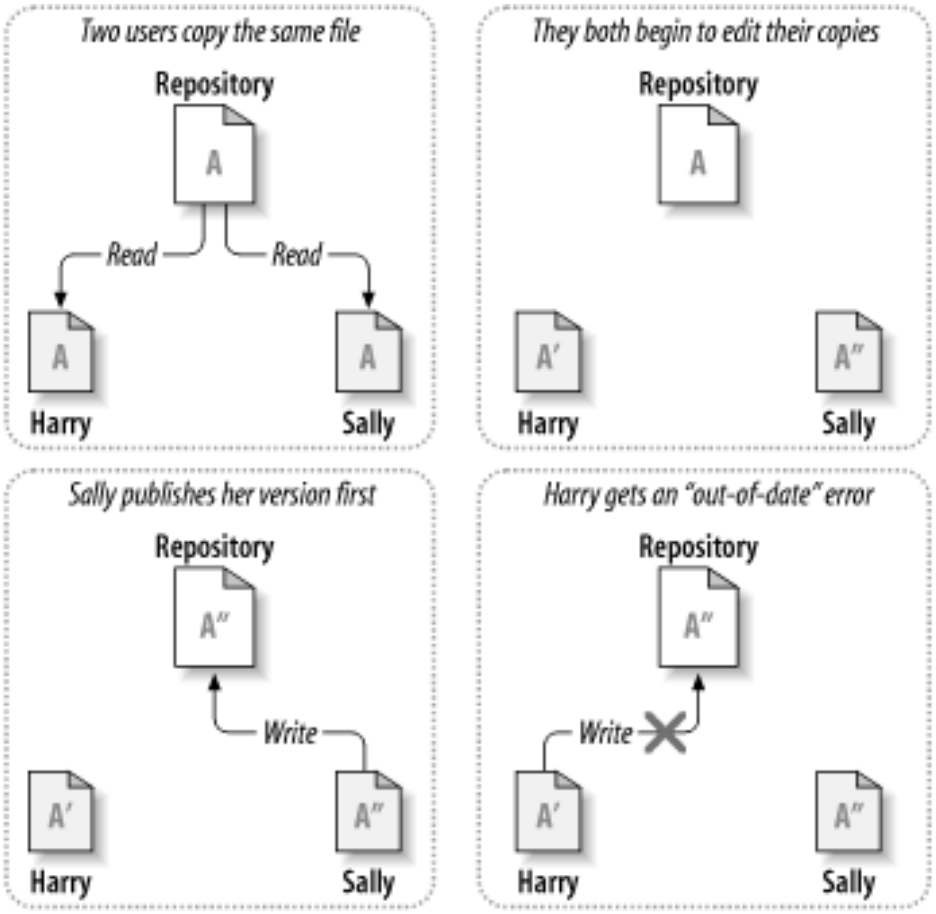
- Provide a central repository for project data
  - Code, documents, other
- Improve development, release, and build management
  - Release cycles
  - Backups
  - Compliance
- Enable productivity – team size, location
- Document software changes
- Track changes
- Revision History
- Undo
- Managed revision control = more efficient file management

Source Control Issue: File Sharing



- Data Overwrite
- Tracking loss
- Wasted time
- Code changes do not match documented changes
- Mass confusion

Source Control Options: #1

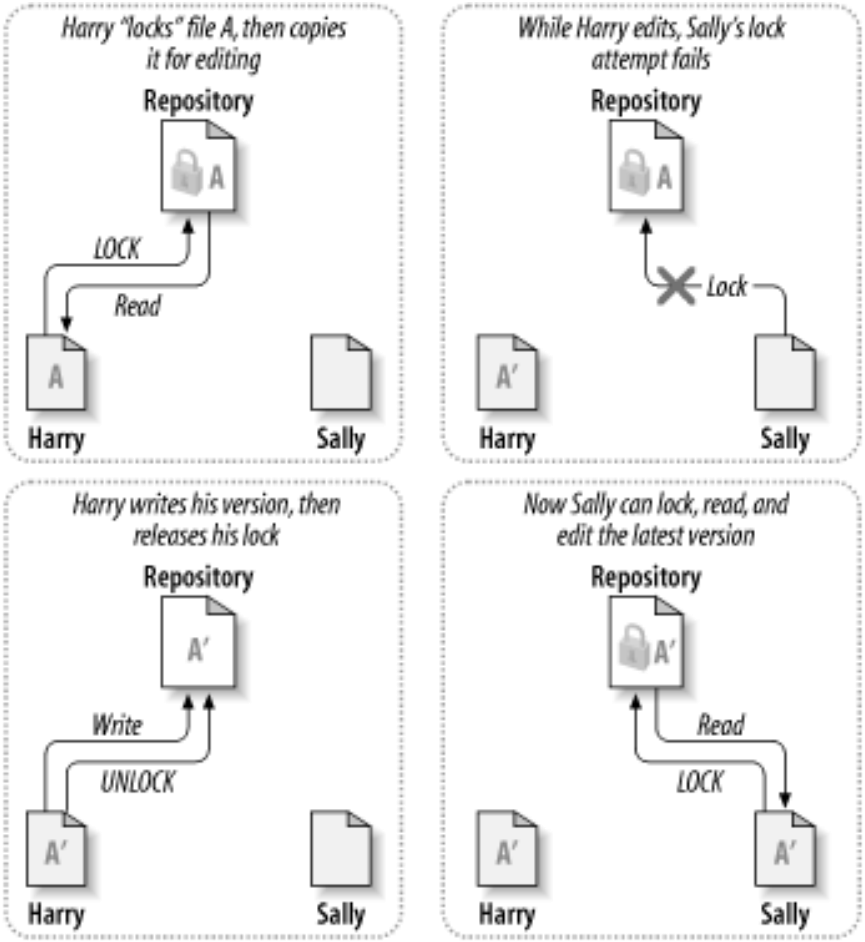


Copy-Modify Method

- Overwrites eliminated
- Tracking loss eliminated
- Less confusion

- Wasted coding
- More frustration – compounded with multiple file edits

Source Control Options: # 2

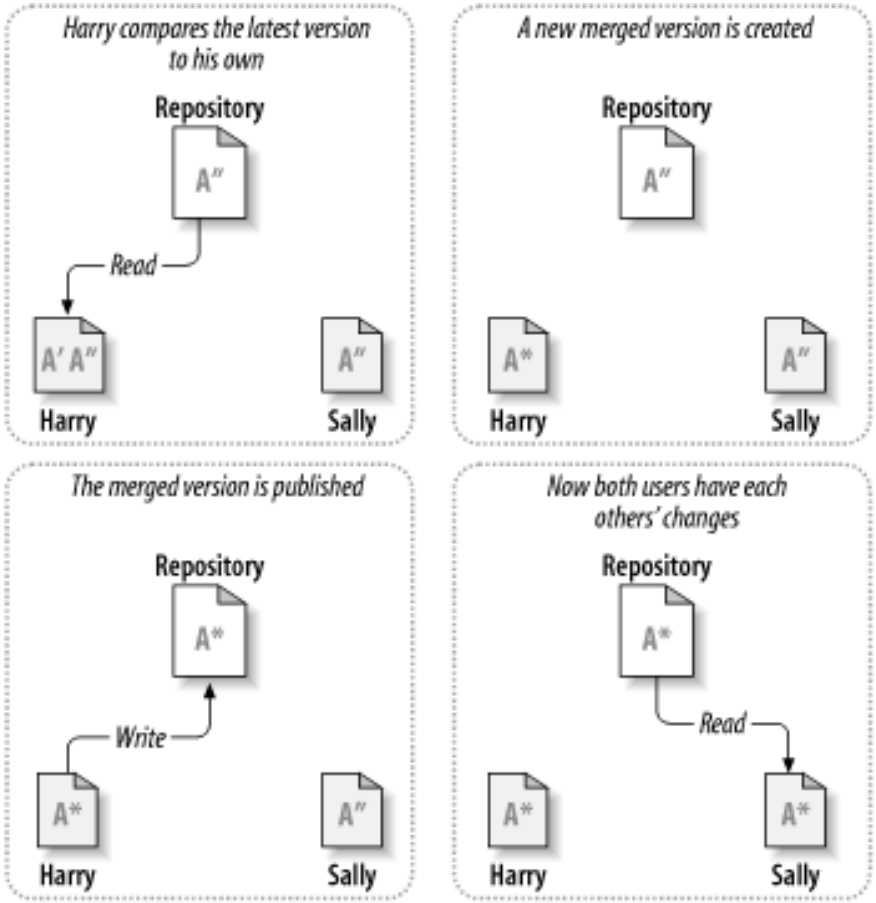


Lock-Unlock Method

- Overwrites eliminated
- Tracking loss eliminated
- Less confusion
- Wasted coding eliminated

- Blocks of files may be locked out simultaneously
- Who has the key?

Source Control Options: # 3



Modify-Merge Method

- Overwrites eliminated
- Tracking loss eliminated
- Less confusion
- Wasted coding eliminated
- Full file access

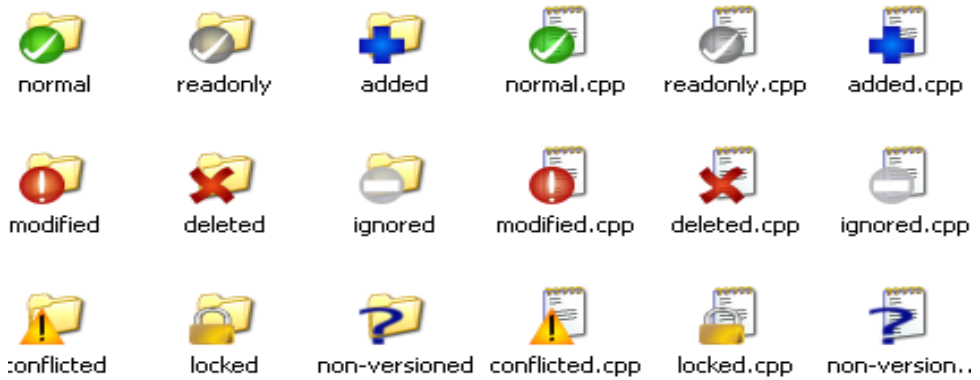
- Are files mergeable?
- Who will merge?

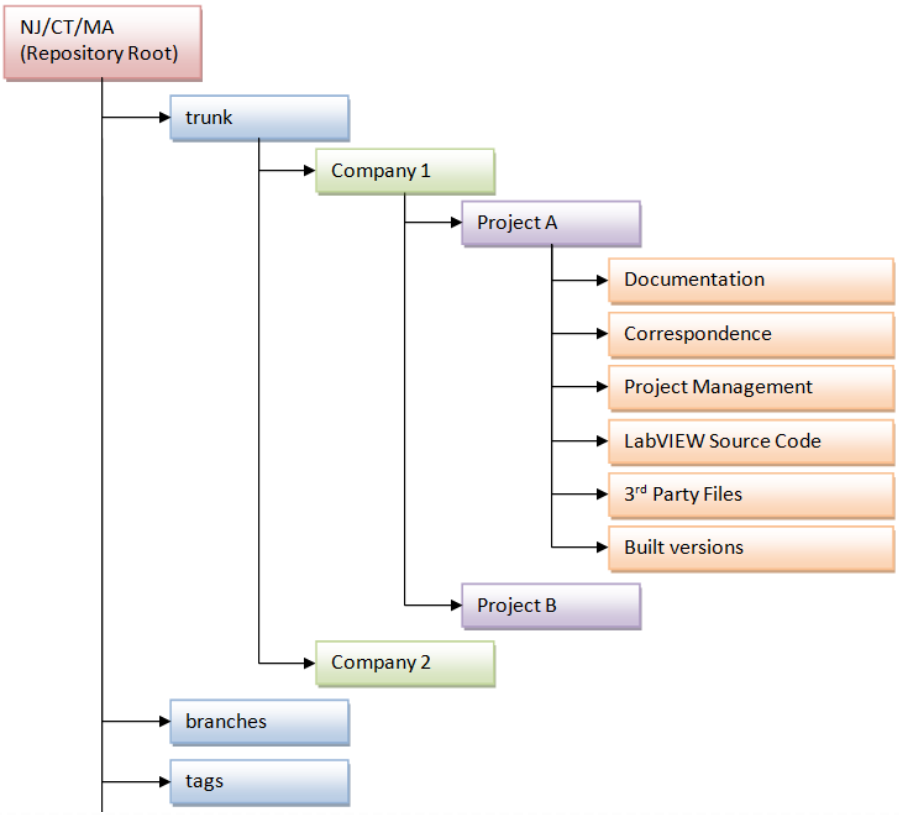
Bloomy Source Code Control

- Copy-Modify-Merge method; File locking as required
  - Caveat: Good planning will localize files under modification
- Subversion Server; TortoiseSVN Client
- Server repository – 1 central store of data
- Working copies – each client has a local copy of the entire repository or logical subsets
- Windows Explorer shell command access
- Icon overlay for status of files in the working copy
- Atomic revisions – “Head” revision is the latest
- Virtual versioned file system
- Branching for separate lines of development

Bloomy Source Code Control

- Windows Explorer environment with icon overlay
- Beyond source code



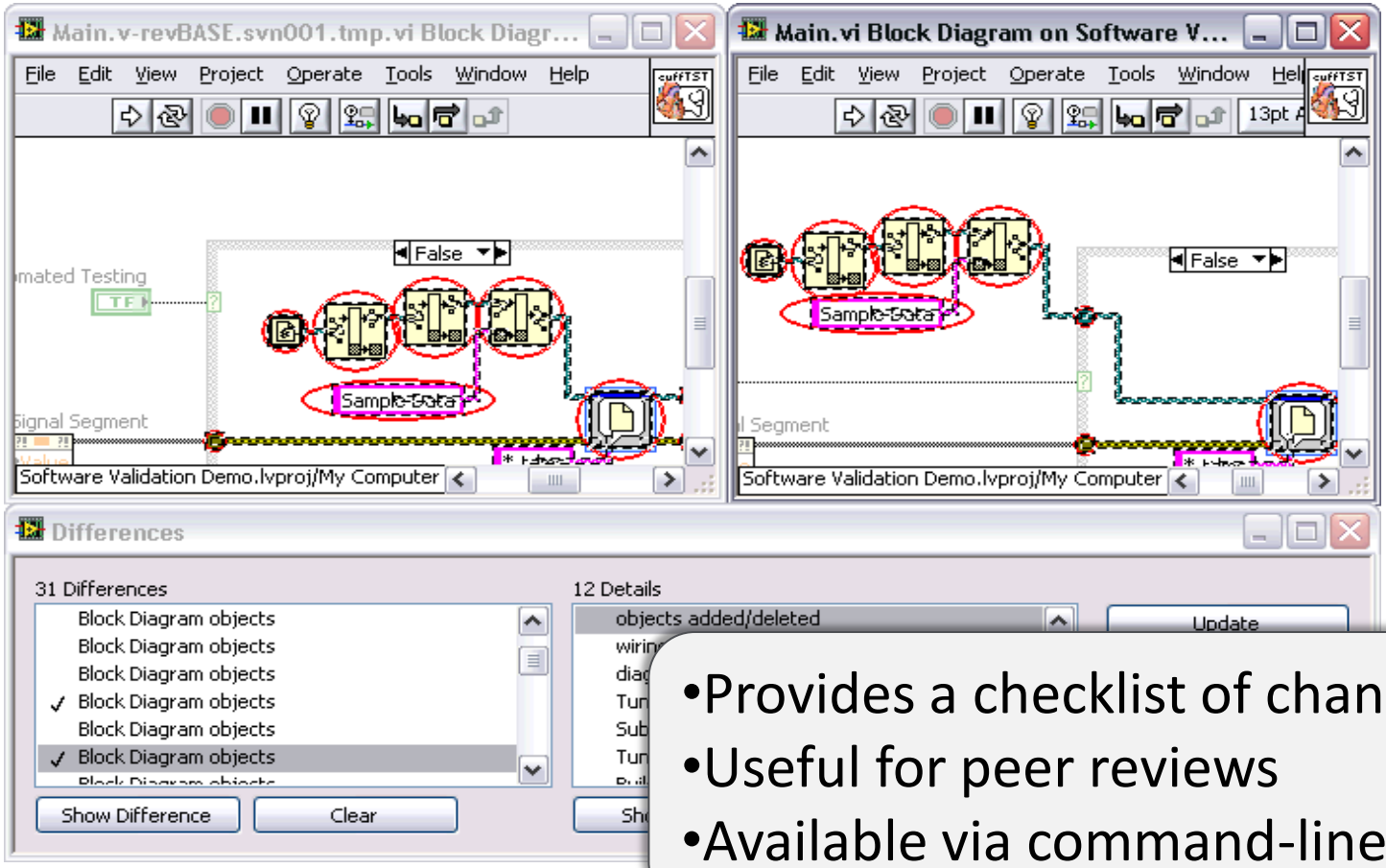


- Standard structure for maintainability
- Internal product lines supported at the ‘company’ level
- Bloomy IP supported at the ‘company’ level
  - Platforms
  - Frameworks
  - Reuse
  - Templates

SCC Still Requires Design

- Good programming style and good system design will ultimately be essential parts of implementing a successful source control
  - Loose coupling
  - Strong cohesion
  - Good project organization
- Design in advance to avoid file sharing / conflict issues
- Strategies / guidelines:
  - Object Oriented Design for creating workable blocks
  - Commit regularly; comment all changes
  - LabVIEW tools to support separation of compiled Code
  - Good communication with co-developers will never go away

Graphing Differences



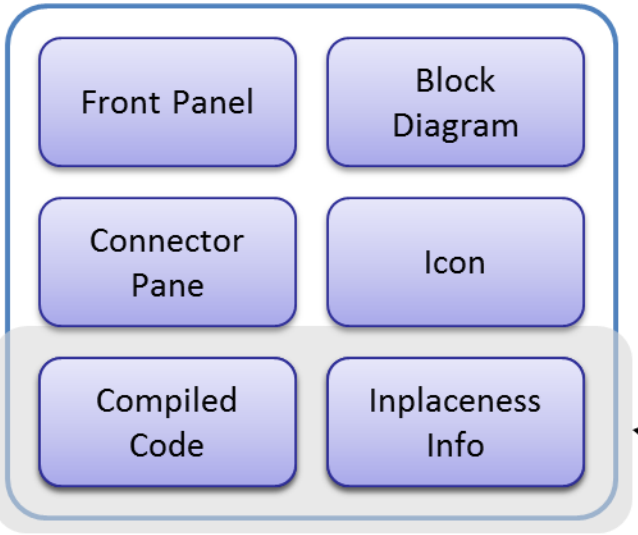
- Provides a checklist of changes
- Useful for peer reviews
- Available via command-line

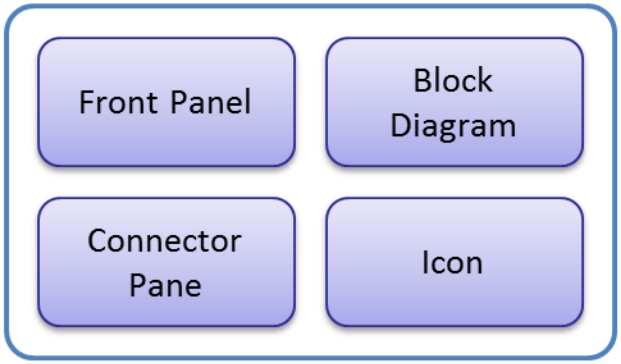
Separate Complied Code from Source File

Improved Source Code Control Integration

2009
.vi file format

.vi file format*





A separate object file is created to store and retain this information

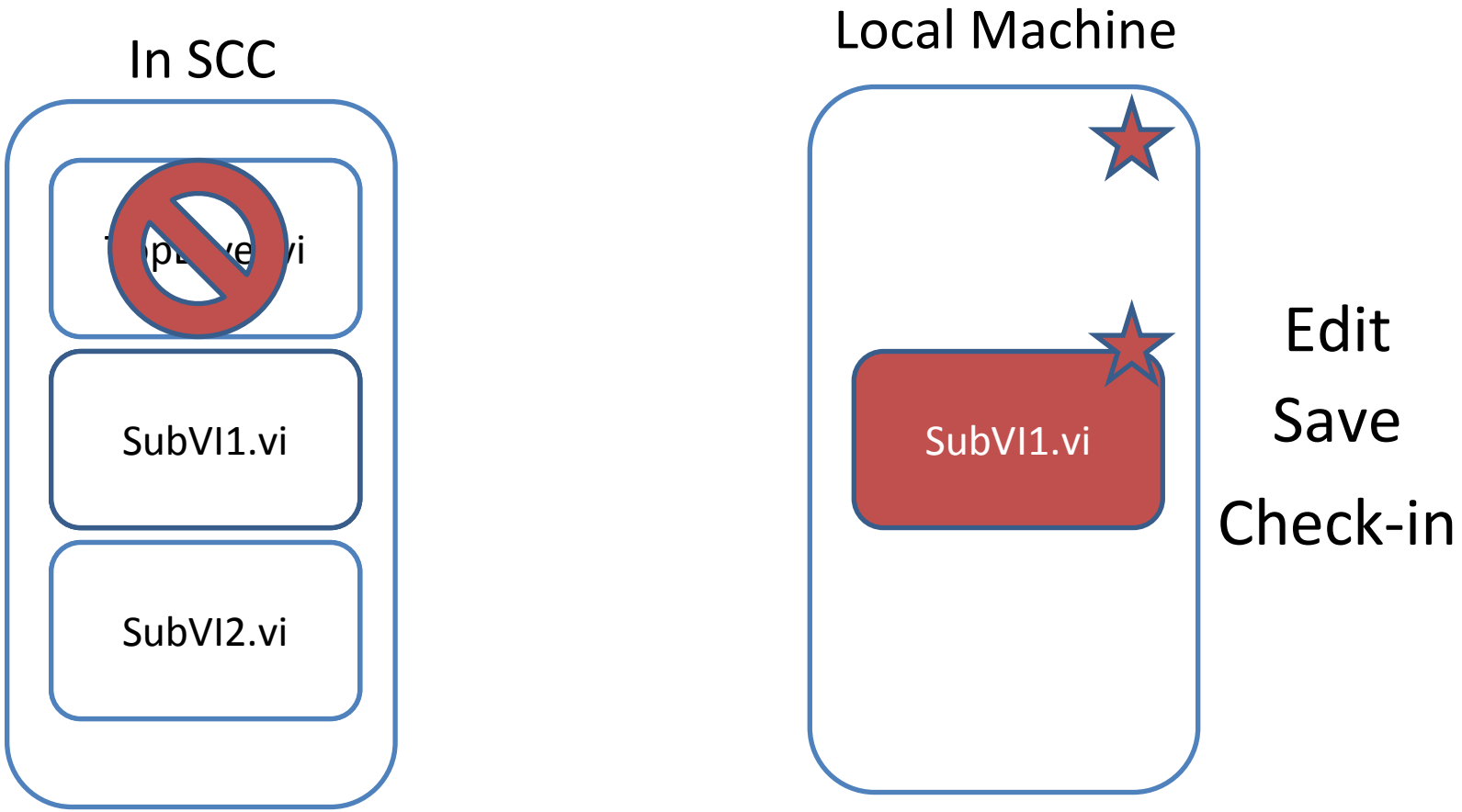
Eliminate the need to re-save and re-submit files to source code control unless the graphical source code has been changed by the developer

**this feature is not on by default and needs to be enabled from the VI Properties dialog*

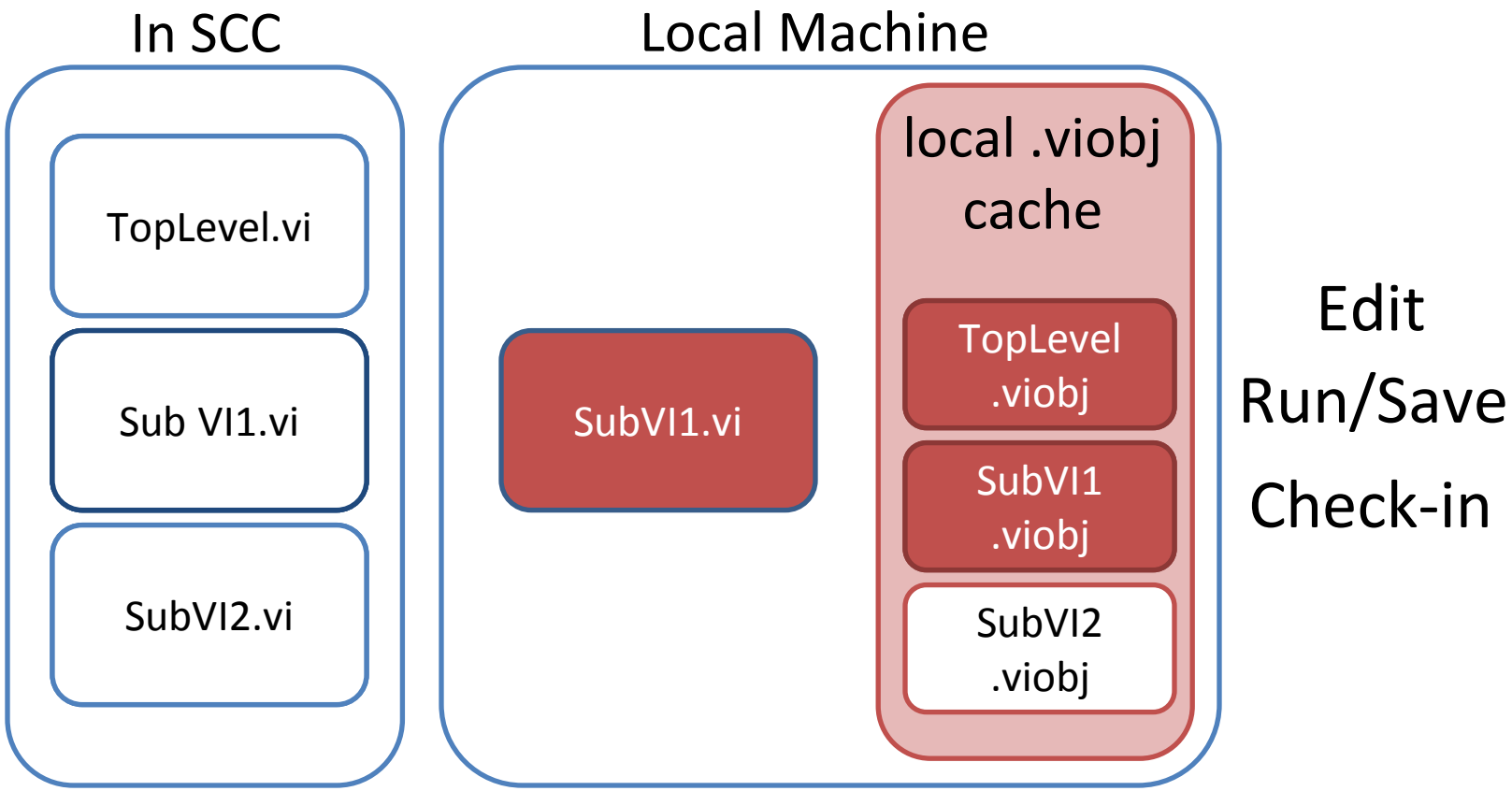
NATIONAL INSTRUMENTS

BLOOMY

Source Code Control Scenario: Pre-2010



Source Code Control Scenario: Post-2010



Demo

Source Code Control

Software Reuse

How to leverage software assets

Software Reuse

- In computer science and software engineering, reusability is the use of existing assets in some form within the software product development process. More than just code, assets are products and by-products of the software development life cycle and include software components, test suites, designs and documentation. -wikipedia-
- Reuse is an integral part of every engineering discipline.
  - Mechanical engineers do not design a combustion engine from scratch for each car rolled off an assembly line
  - Chemical engineers do not develop the formula anew for each bottle of cleaner that is placed on a hardware store's shelf
  - Aerospace engineers do not build solid rocket boosters from ground zero for each space shuttle.
  - Software can also be acquired, developed, maintained, and managed via a "product-line" approach.

(-DOD Software Reuse Initiative)

Why Software Reuse

- Leverage prior investments and productivity to generate:
  - Reduction in labor hours
  - Improvements in schedule
  - Consistency of product
  - Minimization of risk
  - Quality of product
  - Ease of maintainability
- Generate IP
- Collaboration between developers
- Reap benefits of prior investments both internal and external

Source Code Control for SW Reuse

- Subversion comes in handy as a SW Reuse tool
  - Templates
  - VIs
  - Packages
- SW Reuse: Maintain libraries in Subversion; Clients link vi.lib and user.lib folders to reuse library.
  - Separate libraries for LabVIEW versions
  - Commit Monitor SW to keep libraries up to date
- Templates: Maintain in Subversion; Clients use subversion copy and rename function to get started

Demo

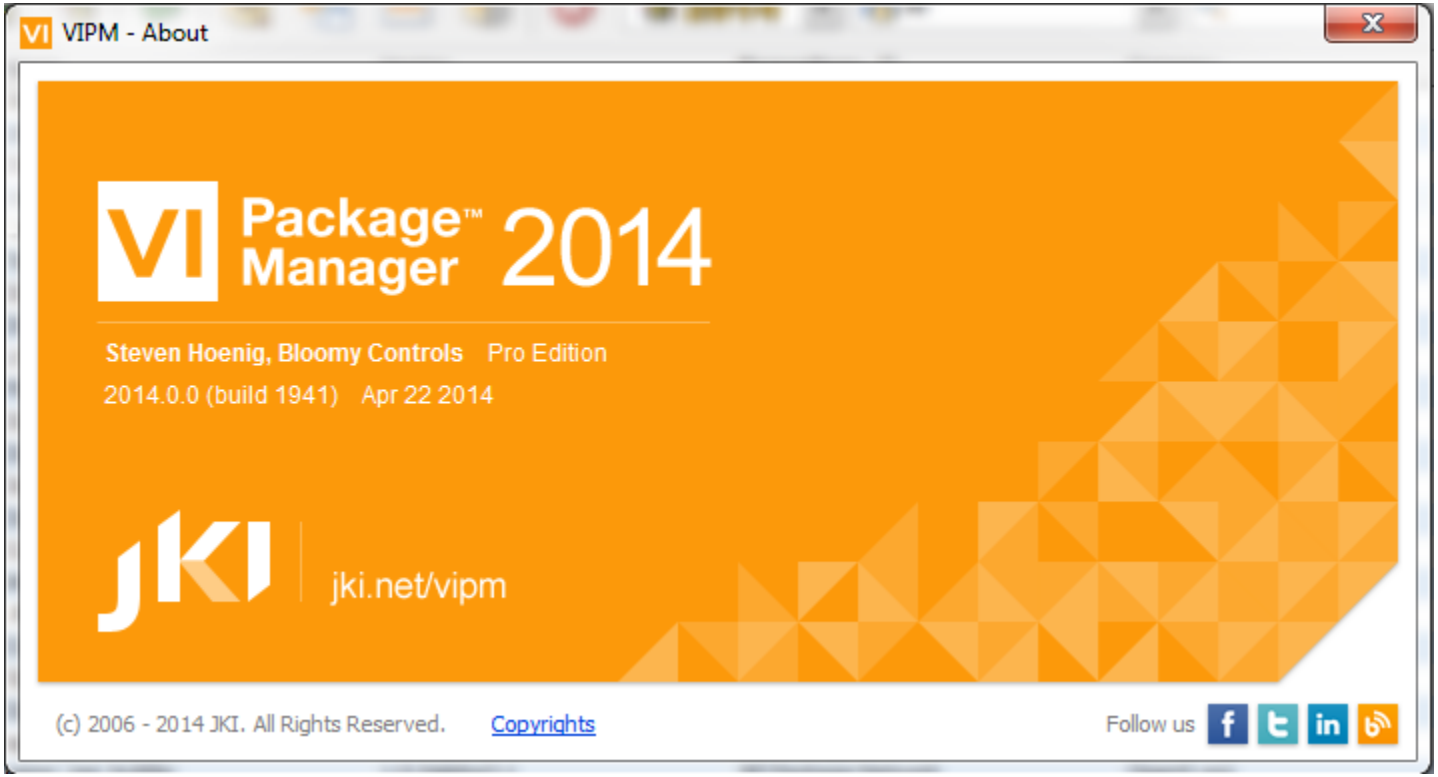
Software Reuse

Reuse Wrap-up, and Next Steps

More Robust Solution Needed for SW Reuse

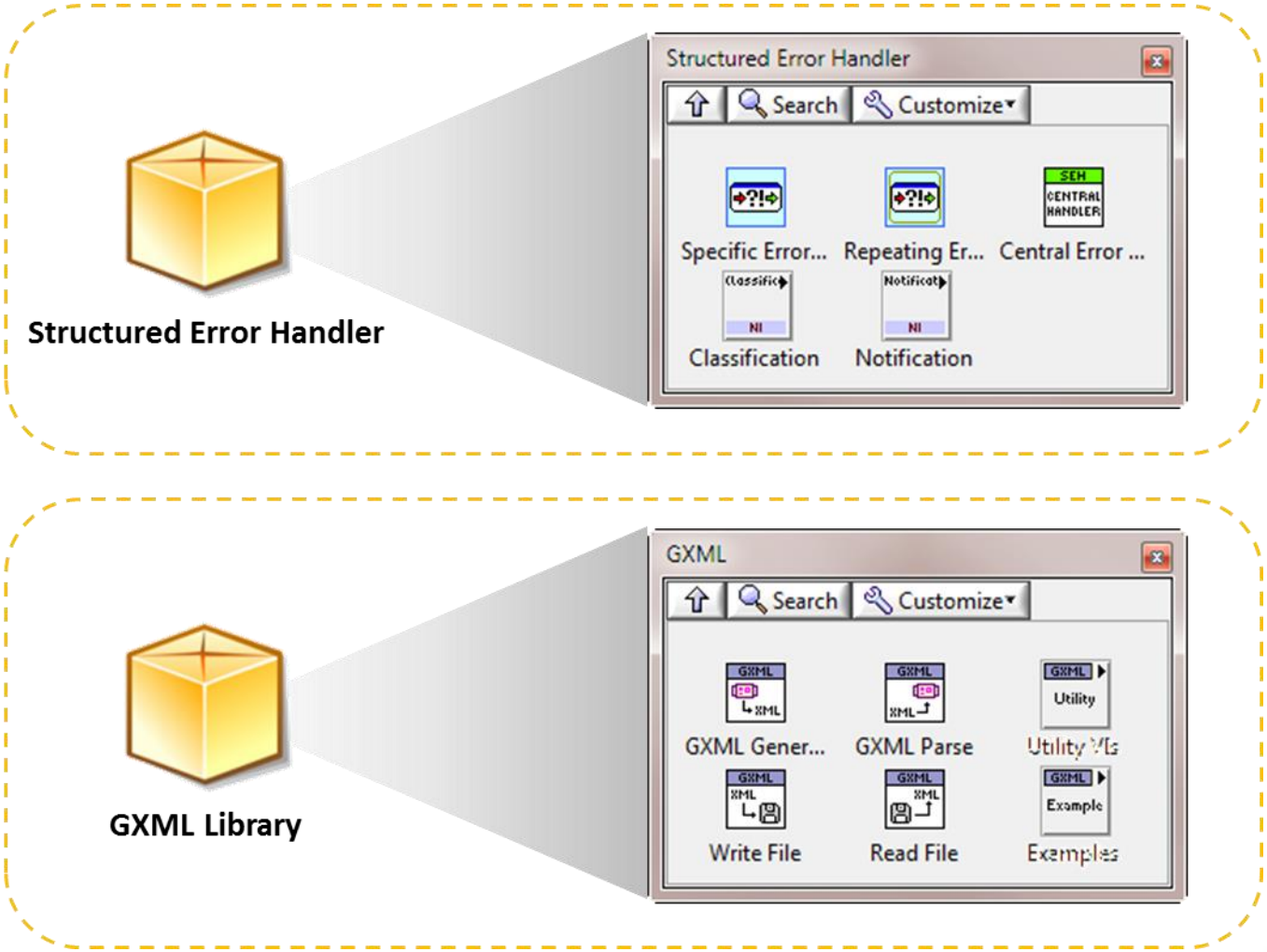
- Ultimately needed to address higher level needs
  - Revisions
  - Single Framework matching openG libraries – one stop shop
  - Supports multiple versions of LabVIEW
  - Professional package for managing/distribution
- Supports upgrades, selective components
- Handles dependency interactions
- Comes with a price-tag for management

VI Package Manager

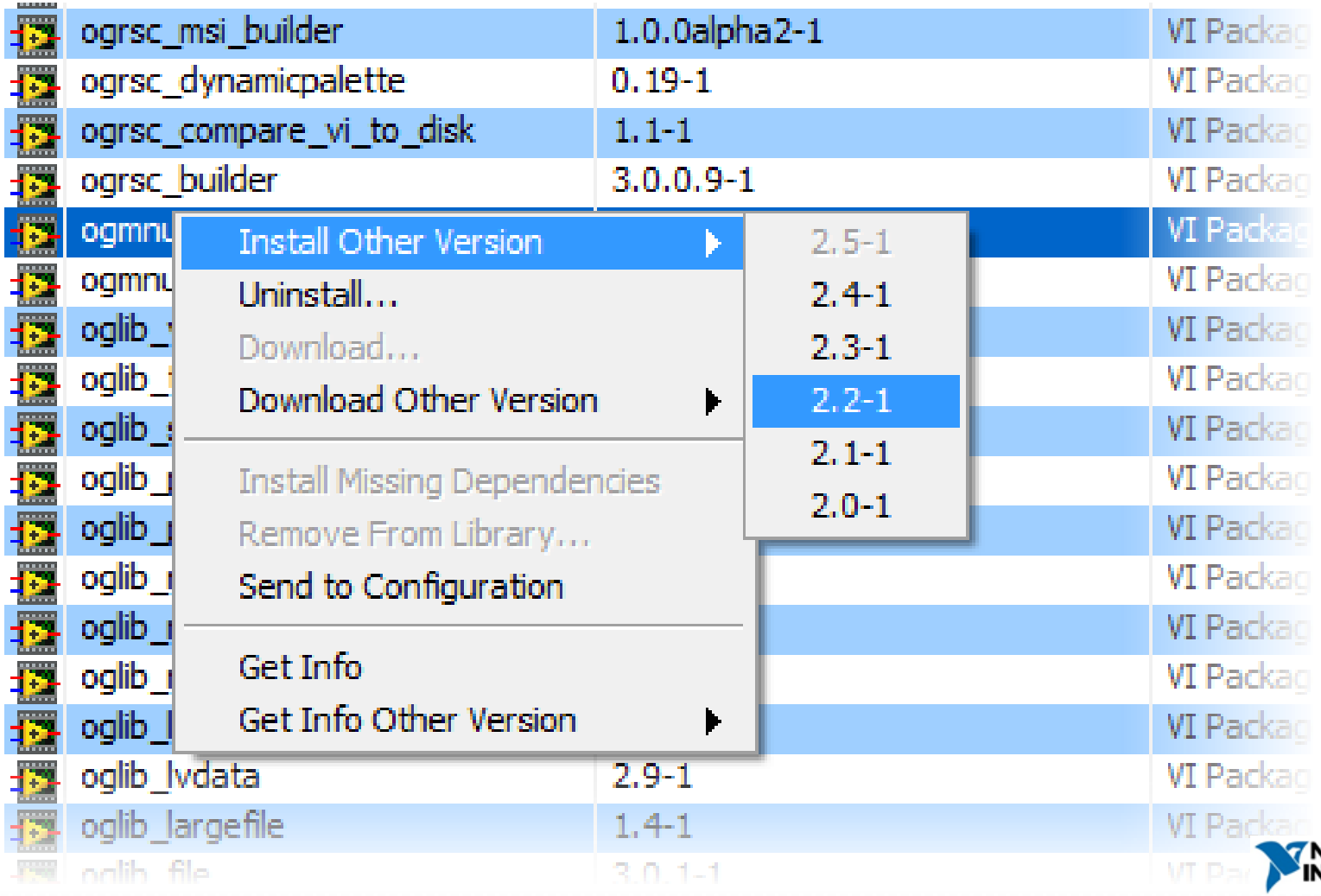


Build and manage packages of LabVIEW code

Install and Manage VI Packages



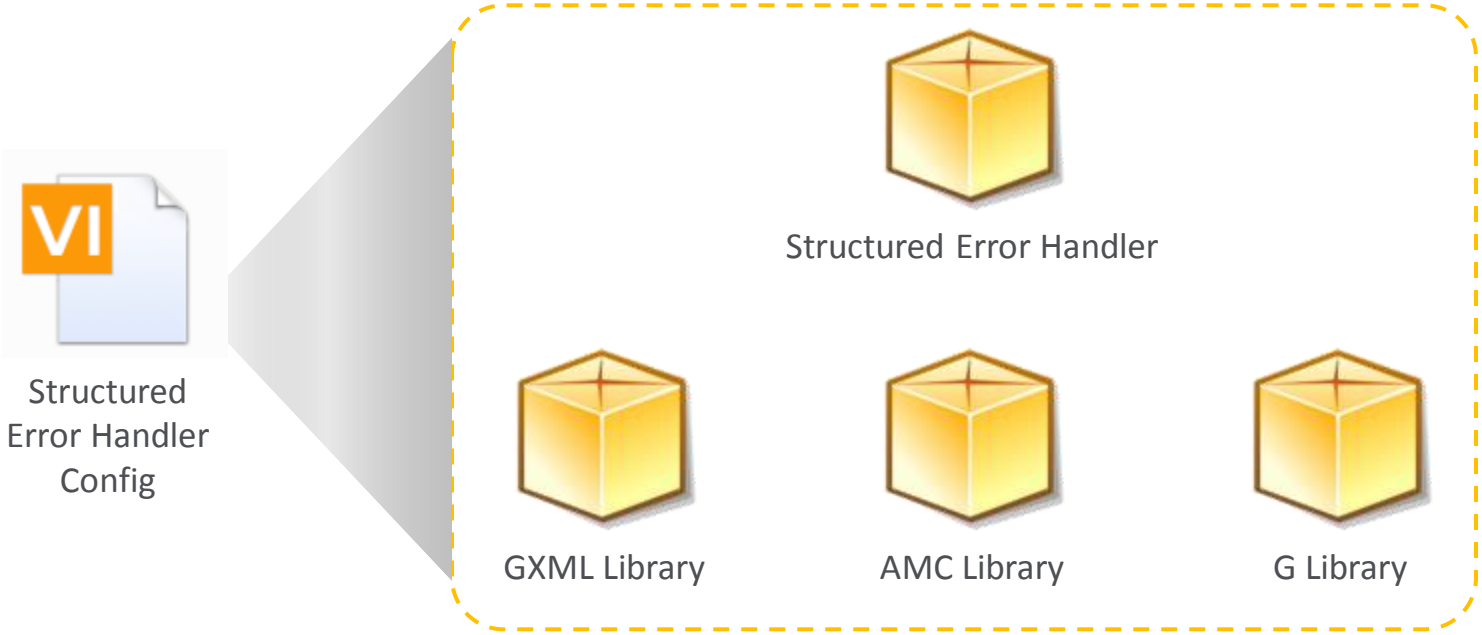
Easily Upgrade and Downgrade Versions



Create VI Configuration Files

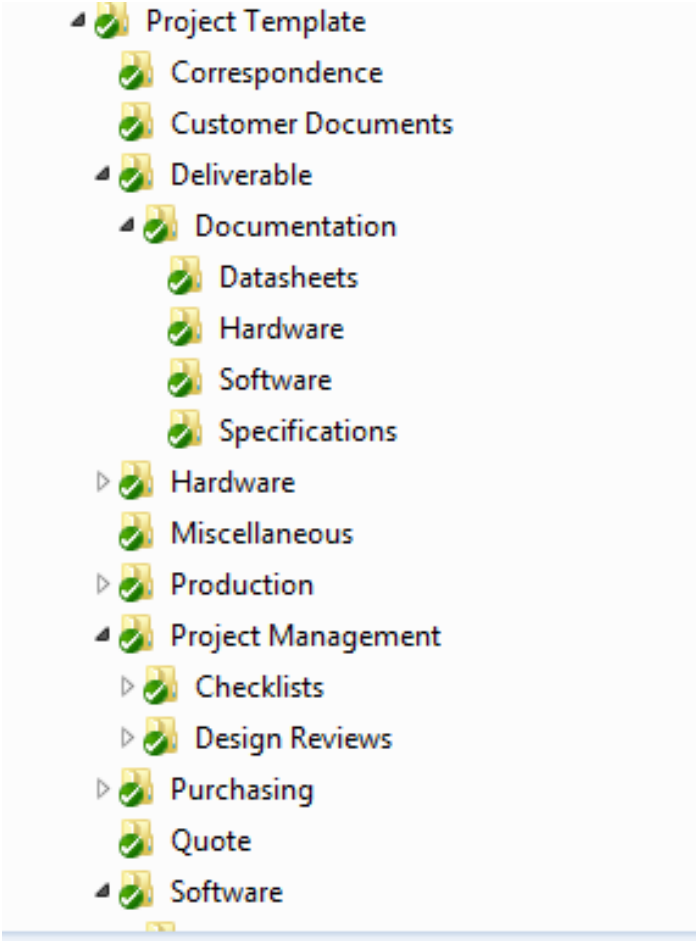
A single file that contains multiple packages

Easily share and distribute code that depends upon multiple libraries



Source Code Control – Broader Impact

- Use Standard Templates to control:
  - Project Structure
  - Source Code Structure
    - Frameworks
    - Design Patterns
  - Deliverables
  - Quality Control
- Configuration Management



Configuration Management

- All project content stored in Subversion
  - Versioning
  - Releases
  - Branching / lockdown
  - Project “snapshot” at any point in the project lifecycle
- Bug Tracking
- Engineering Change Orders
- Traceability to date/time, developer, version

Engineering Change Control

- All project items maintained in Subversion database
  - Source Code
  - Documentation
  - Drawings
  - Communications / meeting minutes
- Update and edit audit trail
- Version control
- Engineering Change Order System

More Information…

Technical White Paper Series
ni.com/largeapps

Online Community Dedicated to Development Best Practices
ni.com/community/largeapps

Bloomy Website
bloomy.com

Contact me at:

Steven Hoenig
Office: 201.773.9115
Mobile: 201.240.8749
steven.hoenig@bloomy.com

ABOUT BLOOMY

Office locations

CONNECTICUT

839 Marshall Phelps Rd.

Windsor, CT 06095-2170

Phone: (860) 298-9925

MASSACHUSETTS

257 Simarano Drive

Marlborough, MA 01752

Phone: (508) 281-8288

NEW JERSEY

39-40 Broadway

Fair Lawn, NJ 07410

Phone: (201) 773-9115

WESTERN U.S. Albuquerque, NM • Phone: (505) 205-2028