Instituto de Computação

Introduction to Version Control

Leonardo Gresta Paulino Murta

leomurta@ic.uff.br

From the beginning…

- 50s
  - Configuration Management used in the production of military airplanes, weapons, and space rockets
- 60s & 70s
  - Born of Software Configuration Management
  - Still focus on military applications
- 80s & 90s
  - Civil uses of Configuration Management (MIL → EIA, IEEE, ISO, etc.)
  - First international standards on Configuration Management
  - Assimilation by non-military organizations

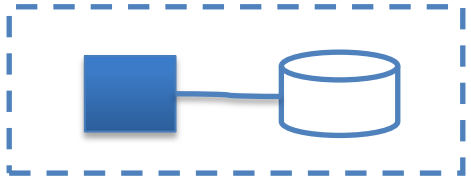
What about Version Control?

Year	System	Highlight
1972	SCCS	First VCS
1982	RCS	First Open Source VCS
1986	CVS as scripts over RCS	First client-server Open Source VCS
1990	CVS rewritten in C	
1992	Rational ClearCase	Market-leader proprietary VCS
2000	Subversion as an evolution of CVS	
	BitKeeper	First (proprietary) VCS to host Linux code
2005	Git	Created by Linus Torvalds
	Mercurial	
	GNU Bazaar	

Obs.: far from being a complete list

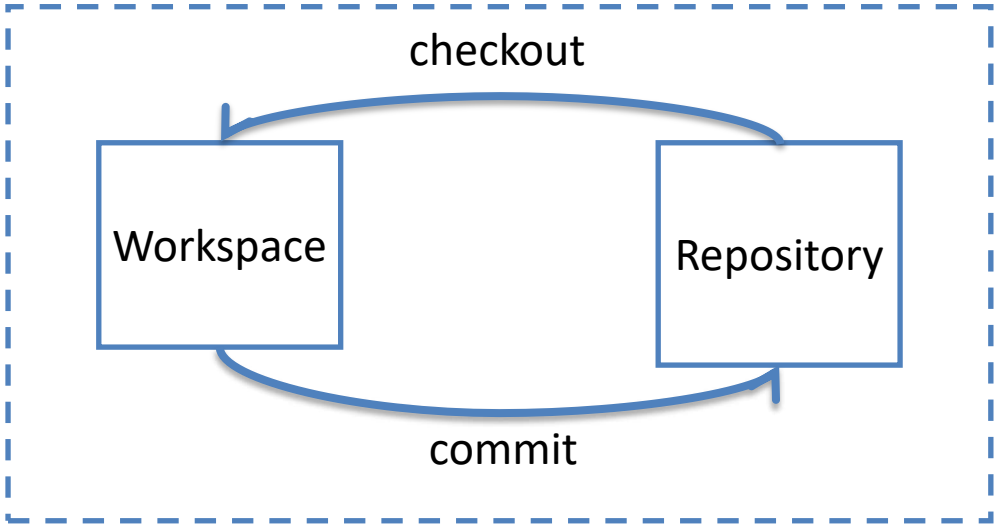
Version Control History

- 70's/80's – local systems
  - SCCS (1972)
  - RCS (1982)



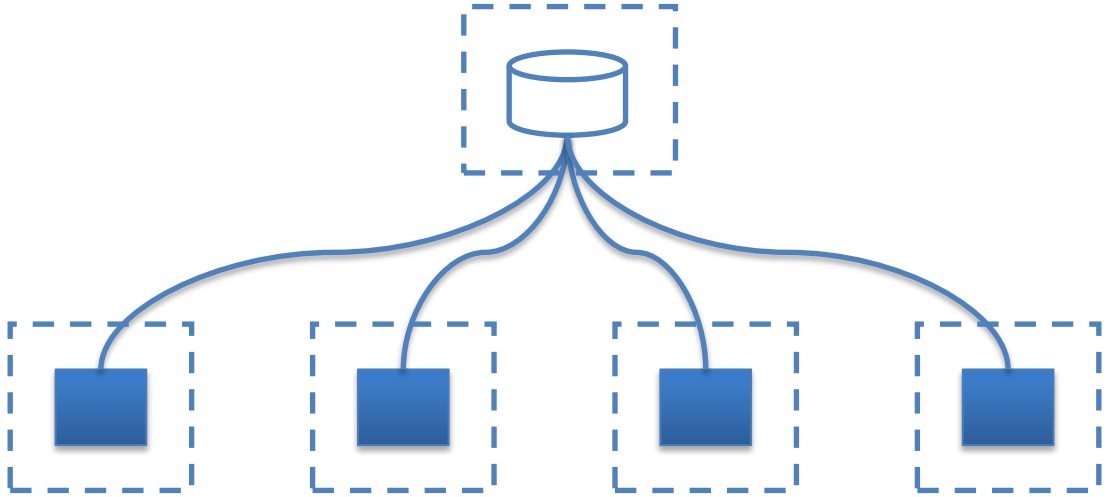
Version Control History

- 70's/80's – local systems
  - SCCS (1972)
  - RCS (1982)



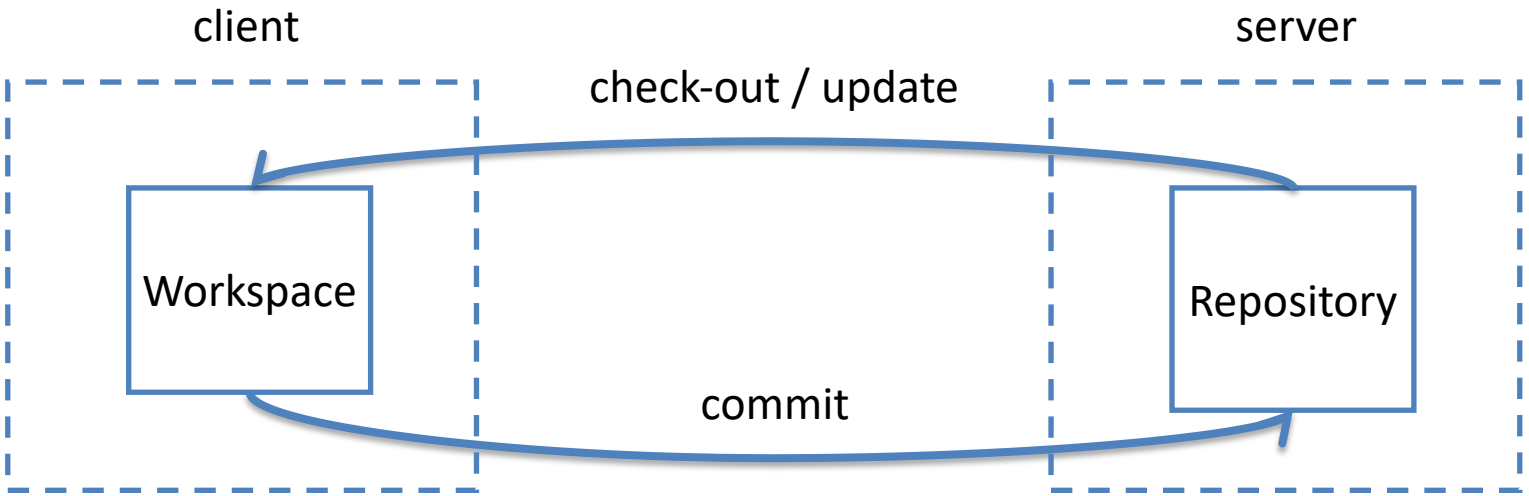
Version Control History

- 80's/90's – client-server systems
  - CVS (1986)
  - Subversion (2000)



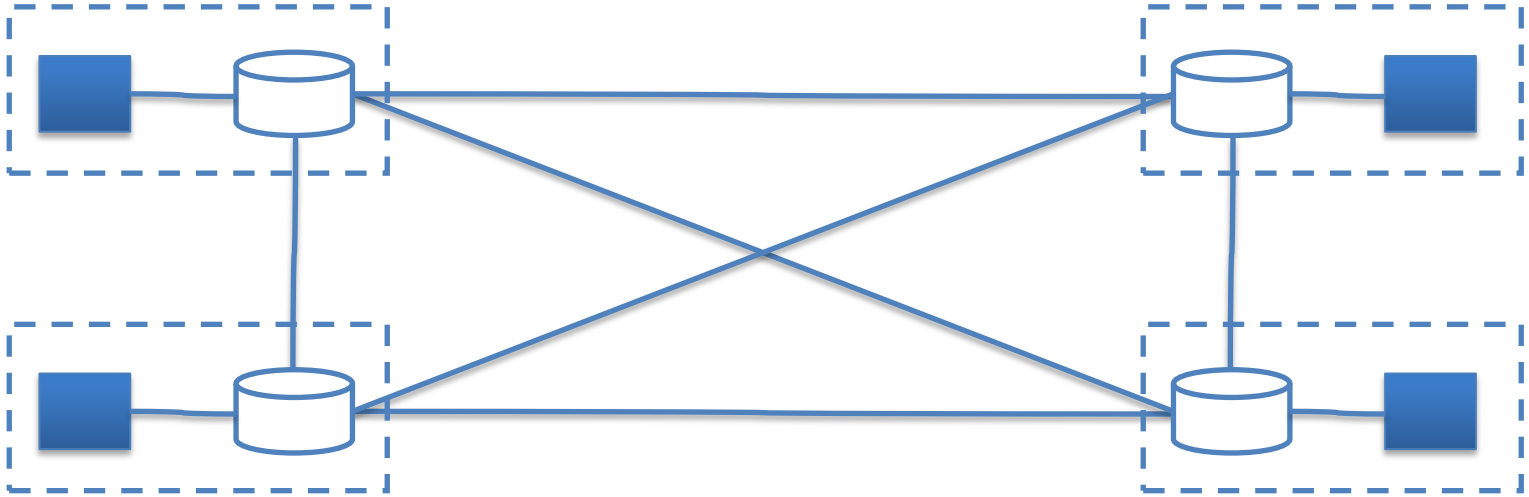
Version Control History

- 80's/90's – client-server systems
  - CVS (1986)
  - Subversion (2000)



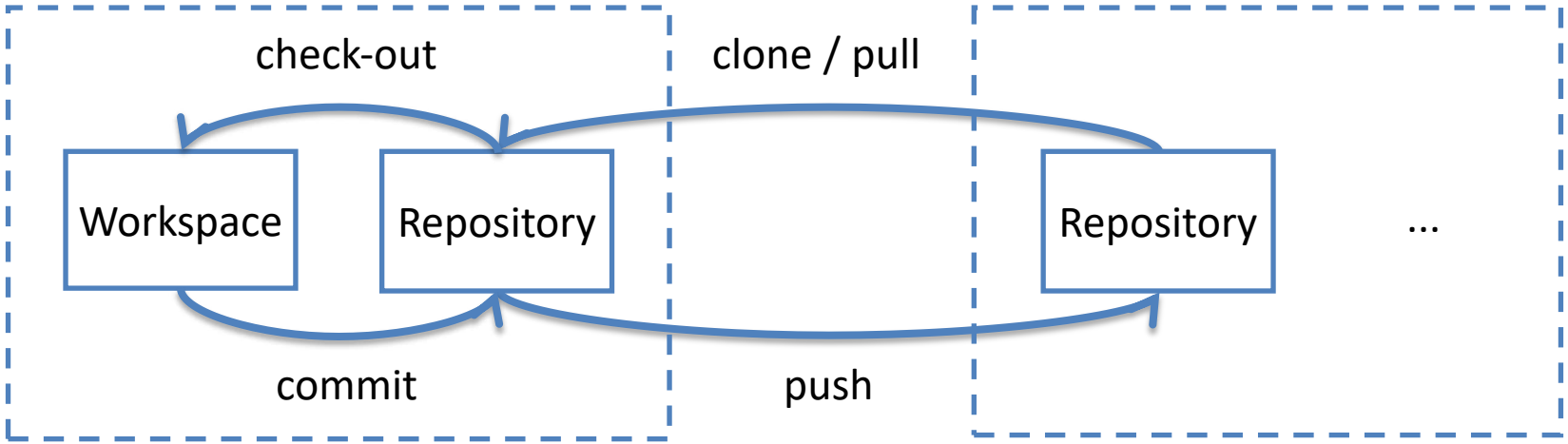
Version Control History

- 2000's – peer-to-peer systems
  - Git (2005)
  - Mercurial (2005)
  - GNU Bazaar (2005)

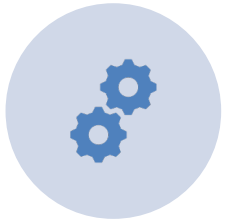


Version Control History

- 2000's – peer-to-peer systems
  - Git (2005)
  - Mercurial (2005)
  - GNU Bazaar (2005)

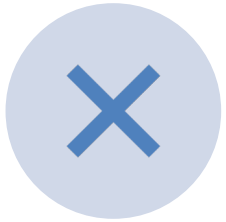


Anti-definitions



Configuration Management is not (only) version control

Version control is not backup



Version control does not avoid changes

Version control is not only for large and complex systems

Version control is not only for large and distributed teams

Forces

- Complex projects in terms of size, sophistication, and technologies
- Huge teams, usually distributed geographically
- Execution requirements for different hardware and software platforms
- Support for different languages
- Different flavors to manage cost/benefit demands (e.g., desktop, standard, professional, and enterprise)

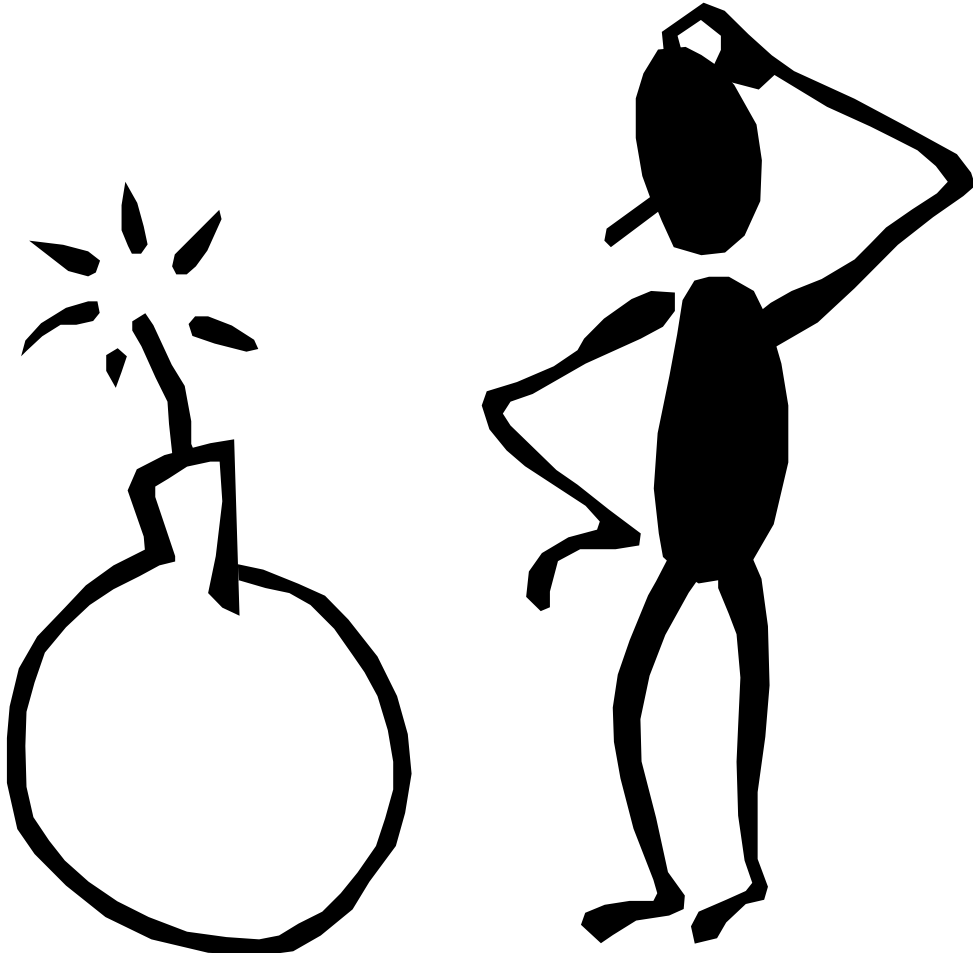
Forces

- Need to reduce the development time to fix defects to comply to Service Level Agreement (SLA) contracts and user expectations

- Necessity to deal with the **inevitable chaos** related to the creative activity of software development, using methods and tools to **maximize productivity** and **minimize defects**

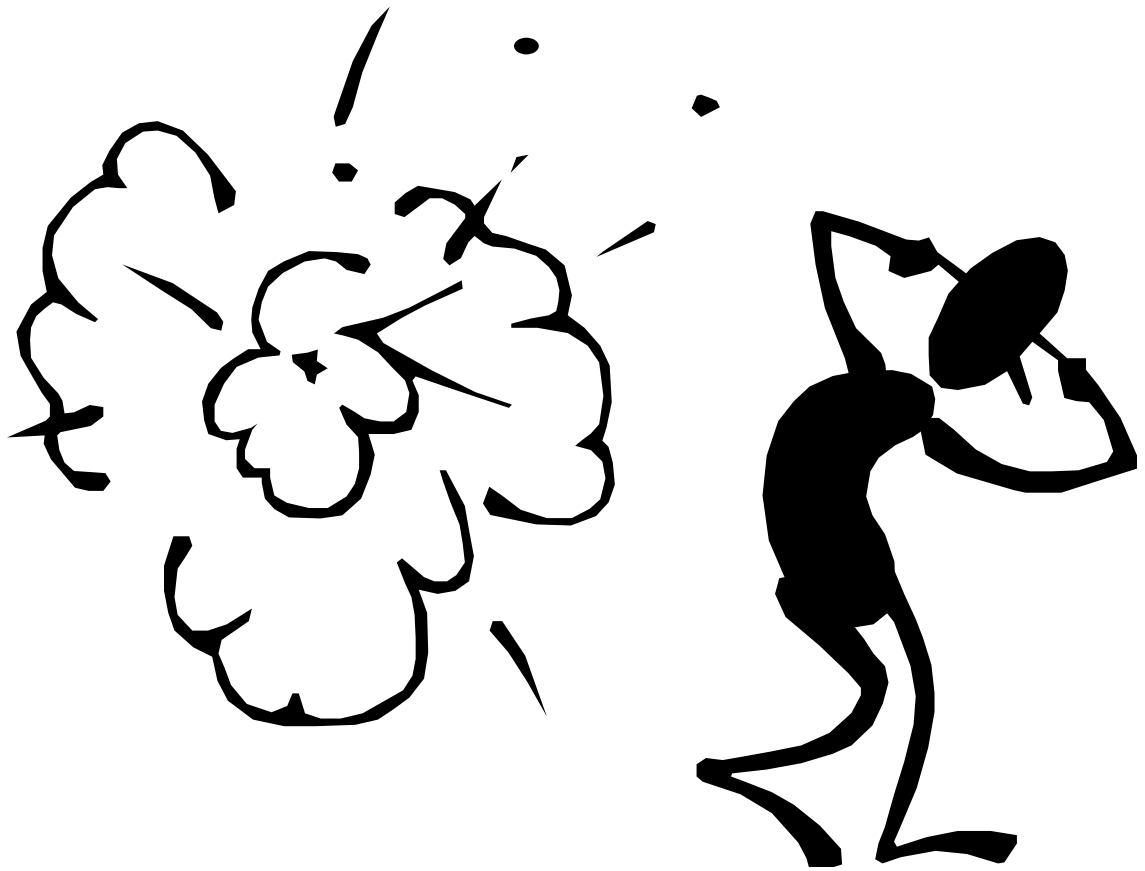
Signals of the absence of Version Control

- Source-code loss
- Libraries fail unexpectedly (dependency hell)
- Difficulty to understand what happened with a program or its parts
- Difficulty to establish when changes were made, why, and by whom



Signals of the absence of Version Control

- Requirements already implemented disappear from the source-code
- Dates in the folder/file names
- Long living commented source-code
- Impossibility to match the executable program version with its source-code version



Some Version Control Benefits

- Increasing organizational memory
- Development more dependent to the process
- Safe parallel development
- Control over the software development and its evolution
- Documentation about the software evolution
- Traceability among versions

References

- Leon A., “Software Configuration Management Handbook”, Artech House, 1st ed., 2004.
- Chacon S., “Pro Git”, 2nd ed., 2014.

Instituto de Computação

Introduction to Version Control

Leonardo Gresta Paulino Murta

leomurta@ic.uff.br