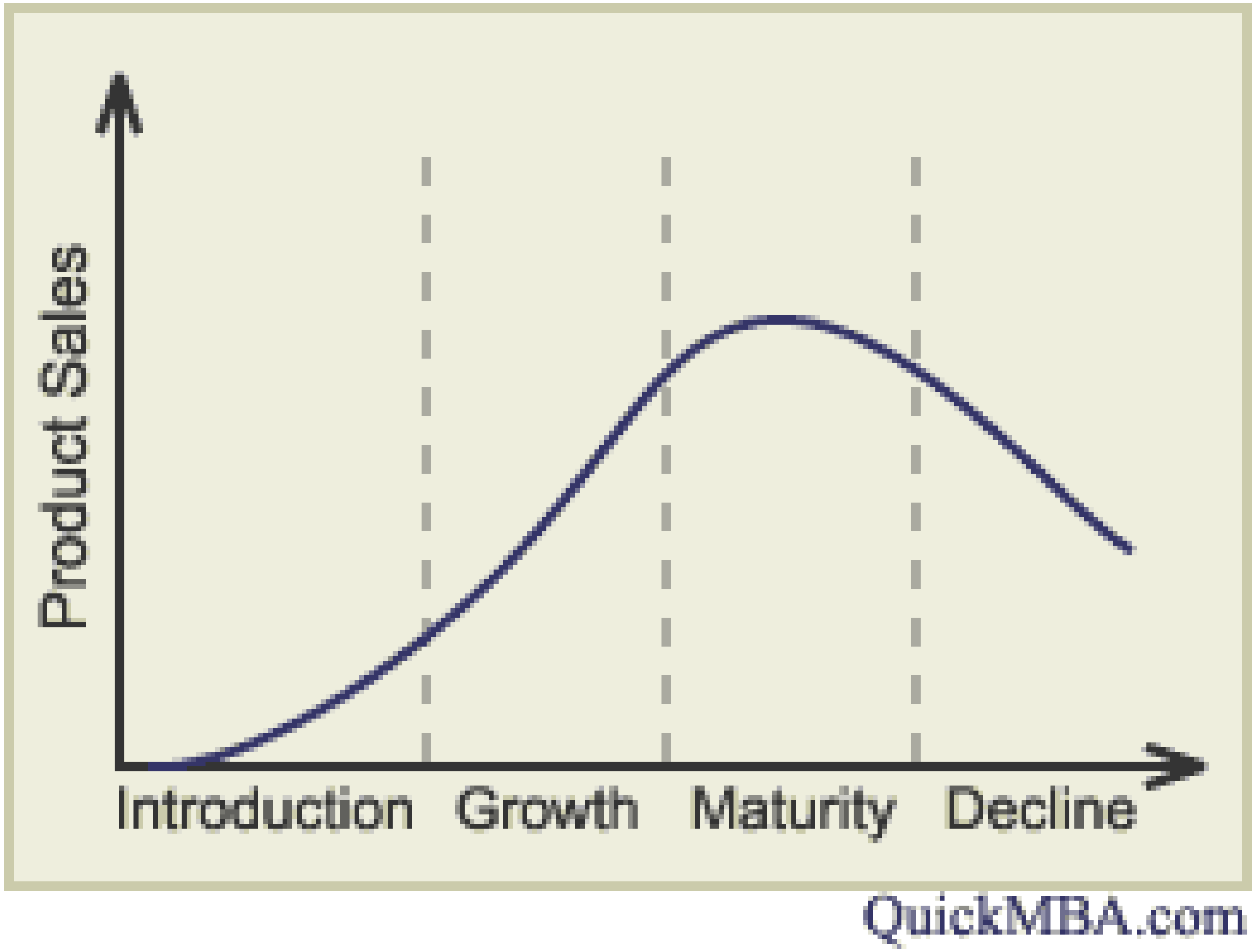
Software Life Span Models

Roberto A. Bittencourt
Based on Rajlich's slides

Software life span models

- Stages through which software goes, from conception to death
- Stages may be very different
- Software = product
  - stages are similar to the stages in the life span of other products

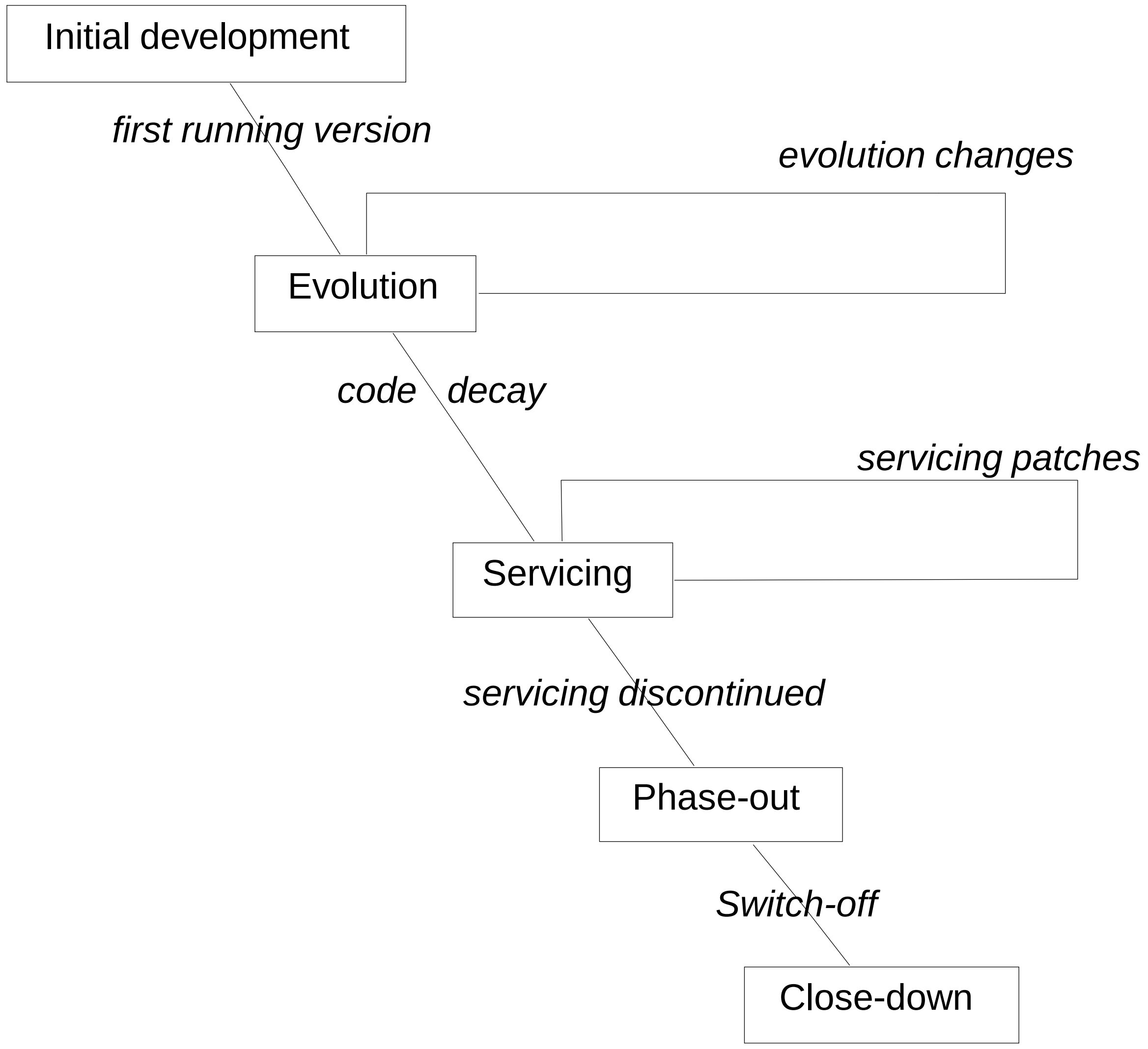
Product lifespan



Software lifespan

- Software is a product
  - sales go through the same lifespan
- Unique proprietary software
  - value follows the same curve
- Names of stages are different

Staged model



Initial development

- Requirements
- Design
- Implementation
  - similar to waterfall, but of limited duration

- Fundamental decisions
  - technology
    - programming language, coding conventions, libraries,…
  - architecture
    - components, interactions
  - program domain knowledge
    - the knowledge is required for evolution

Evolution

- Adapts the application to the ever-changing user and operating environment
- Adds new features
- Corrects mistakes and misunderstandings
- Responds to both developer and user learning
- Program usually grows during evolution
- Both software architecture and software team knowledge make evolution possible

Code decay

- Loss of software coherence
- Loss of the software knowledge
  - less coherent software requires more extensive knowledge
  - if the knowledge is lost, the changes will lead to a faster deterioration
- Loss of key personnel = loss of knowledge
- Challenge: eliminate or slow code decay

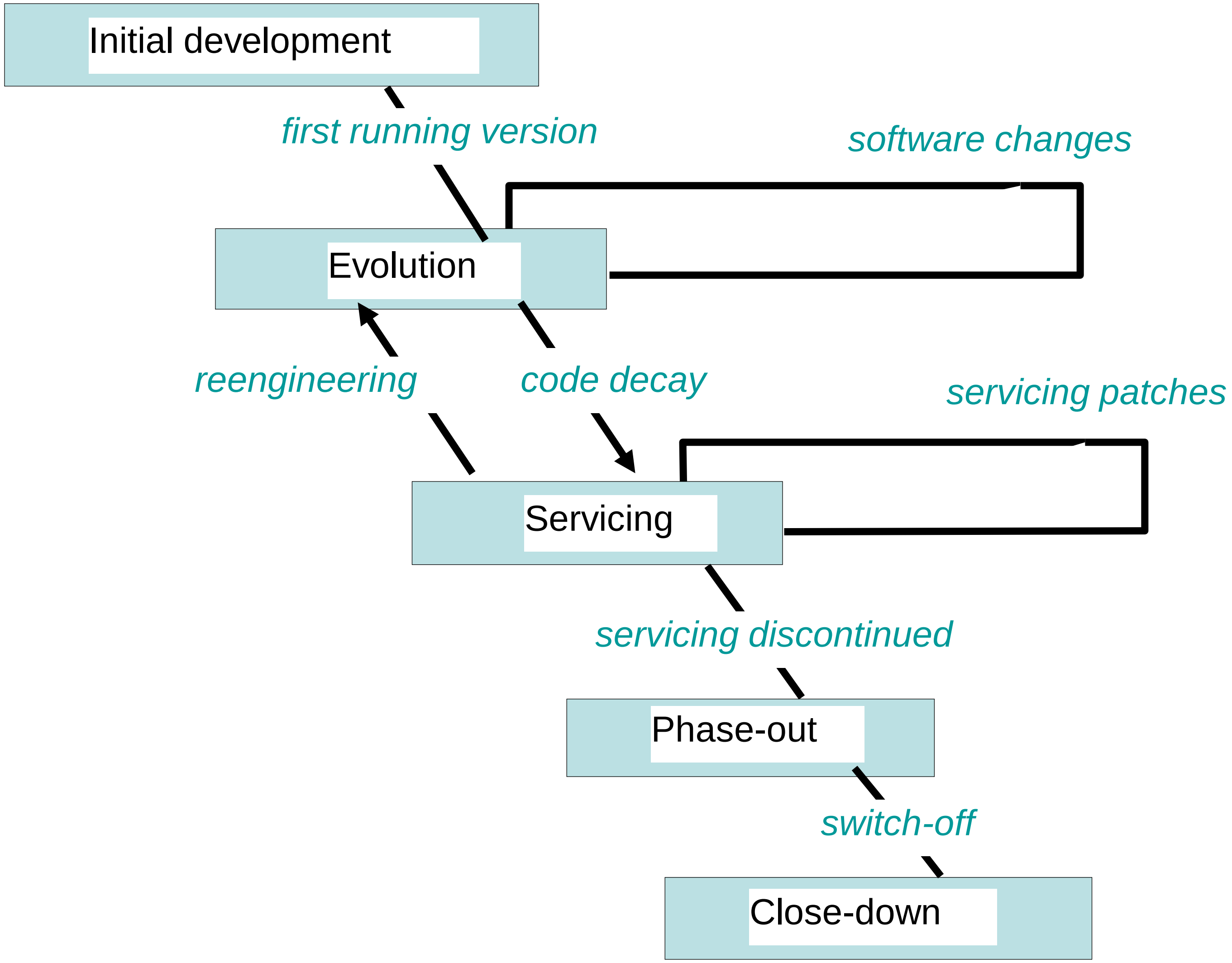
Servicing

- The program is no longer evolved
  - it either decays or stabilizes or managers decide not to support evolution
- Changes are limited to patches and wrappers
  - less costly, but they cause further deterioration
- Process is very different from evolution
  - no need for senior engineers
  - the process is stable
    - well suited to process measurement and management

Reversal from servicing to evolution

- Very expensive, rare
- Not simply a technical problem
  - the knowledge of the software team must also be addressed

Reengineering



Phase-out

- No more servicing is being undertaken
  - but the system still may be in production
- Users must work around known deficiencies

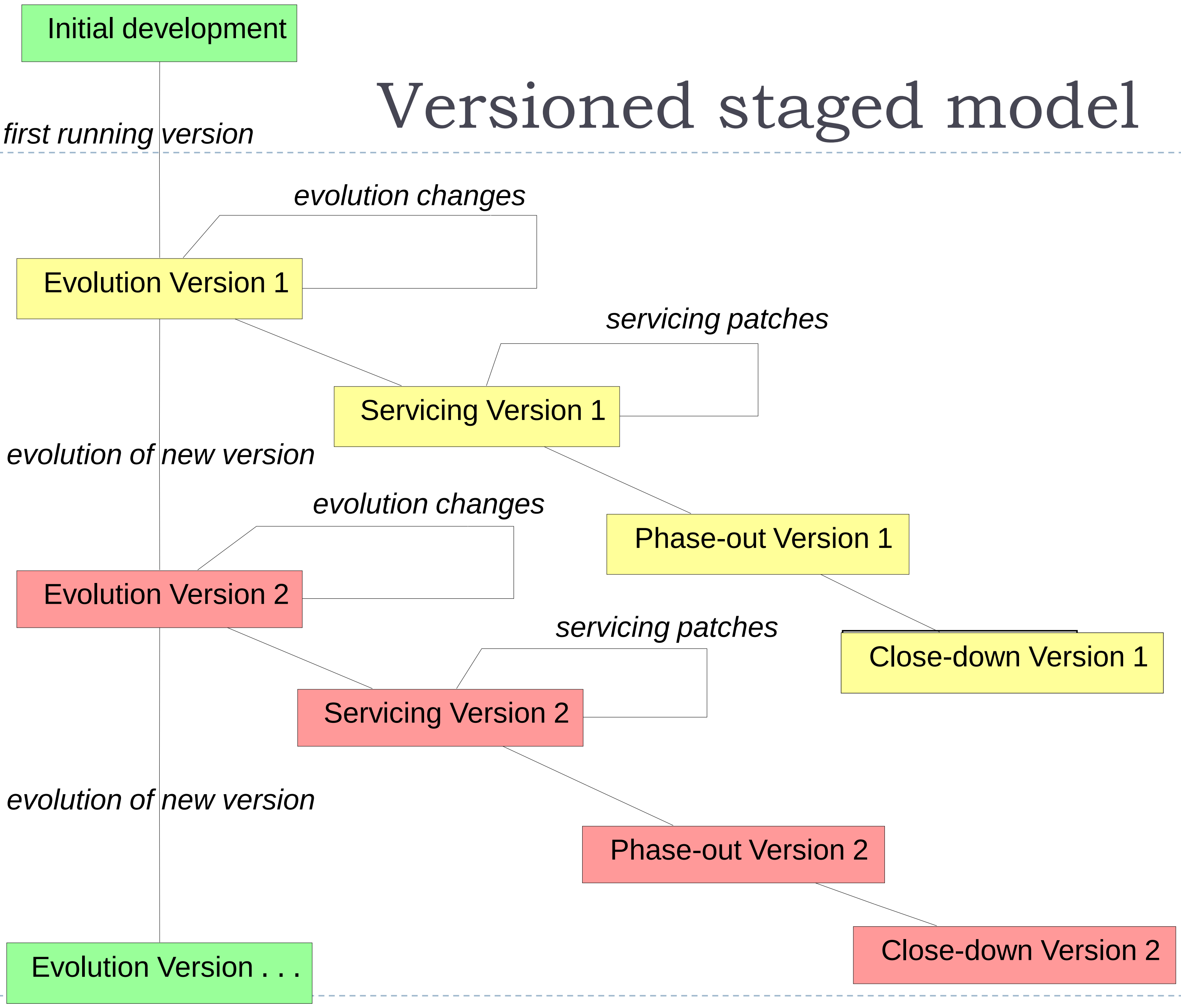
Close-down

- Software use is disconnected
  - current life of successful software:
  - about 10 to 20 years
- Users are directed towards a replacement
- An ‘exit strategy’ is needed.
  - changing to another system requires retraining
  - what to do with long-lived data?

Versioned staged model

- Used by software with many users
- Evolution is the backbone of the process
  - evolution produces versions
  - versions are serviced, phased-out, closed down

Versioned staged model



Mozilla Firefox releases

2.0	3.0	3.5	version #	Date
x			2.0.0.12/	2/7/2008
	x		3.0b3/	2/13/2008
	x		3.0b4/	3/11/2008
x			2.0.0.13/	3/25/2008
	x		3.0b5/	4/9/2008
x			2.0.0.14/	4/15/2008
	x		3.0rc1/	5/15/2008
	x		3.0rc2/	6/4/2008
	x		3.0rc3/	6/11/2008
	x		3.0/	6/19/2008
x			2.0.0.15/	6/23/2008
x			2.0.0.16/	7/11/2008
	x		3.0.1/	7/16/2008
x			2.0.0.17/	9/17/2008
	x		3.0.2/	9/22/2008
	x		3.0.3/	10/7/2008
x			2.0.0.18/	11/11/2008
	x		3.0.4/	11/11/2008
x			2.0.0.19/	12/15/2008
	x		3.0.5/	12/15/2008
x			2.0.0.20/	12/18/2008
	x		3.0.6/	2/2/2009
	x		3.0.7/	3/3/2009
	x		3.0.8/	3/27/2009
	x		3.0.9/	4/9/2009
		x	3.5b4/	4/24/2009
	x		3.0.10/	4/27/2009
		x	3.5b99/	6/7/2009
	x		3.0.11/	6/10/2009
		x	3.5rc1/	6/16/2009
		x	3.5rc2/	6/17/2009
		x	3.5rc3/	6/24/2009
		x	3.5/	7/1/2009
		x	3.5.1/	7/17/2009
	x		3.0.12/	7/20/2009
		x	3.5.2/	7/30/2009
	x		3.0.13/	7/31/2009
		x	3.5.3/	8/24/2009
	x		3.0.14/	9/8/2009
		x	3.5.4/	10/19/2009
	x		3.0.15/	10/26/2009

- 2008 – 2009
- Versions 2.0 and 3.0
  - serviced in parallel
- Version 3.5 introduced 4/2009
  - while version 3.0 still serviced
  - while version 2.0 in phase-out

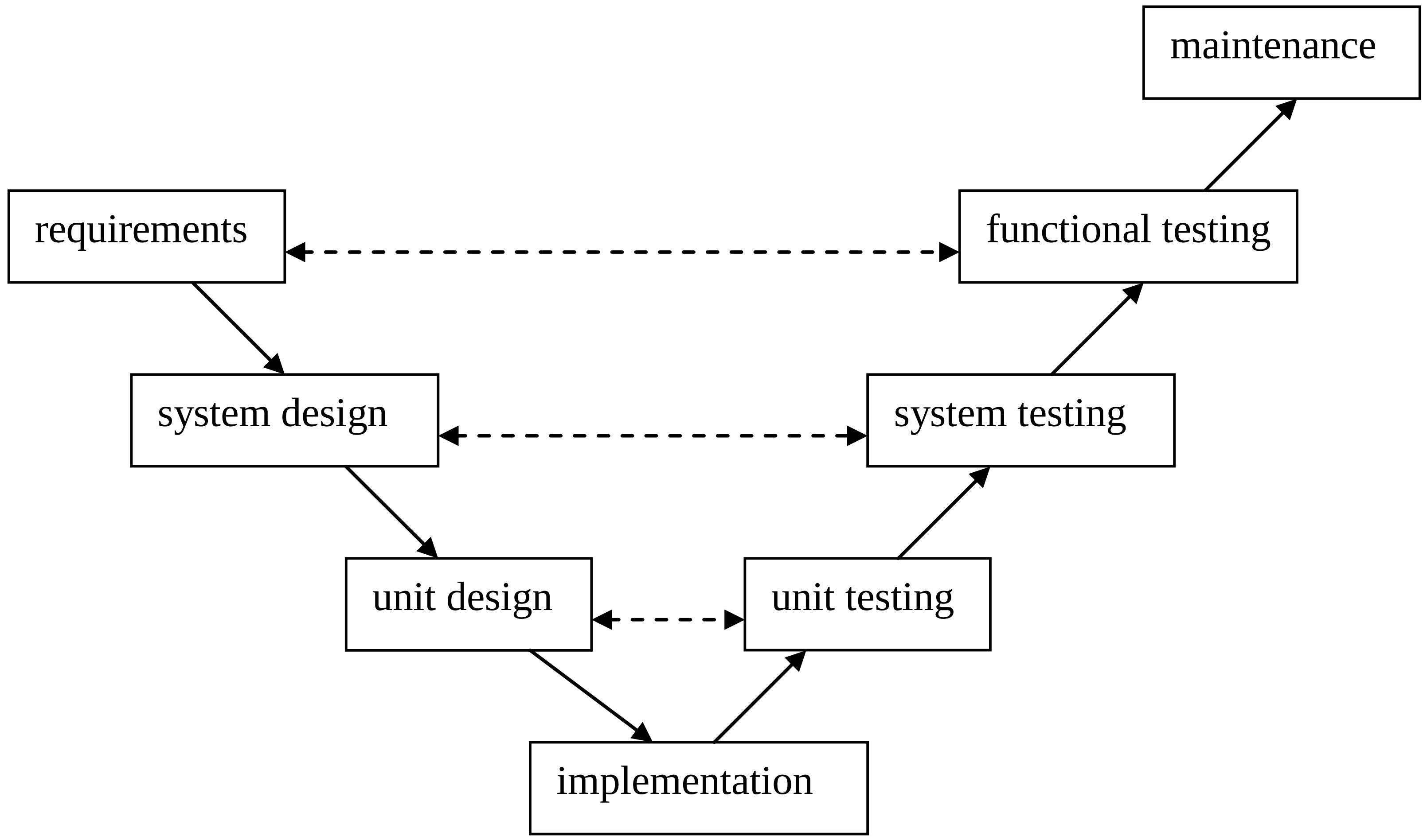
Incomplete lifespans

- Discontinued projects
  - stopped during initial development
- Stable domain
  - no need for evolution
- Development starts with evolution
  - a related old software is evolved into new one

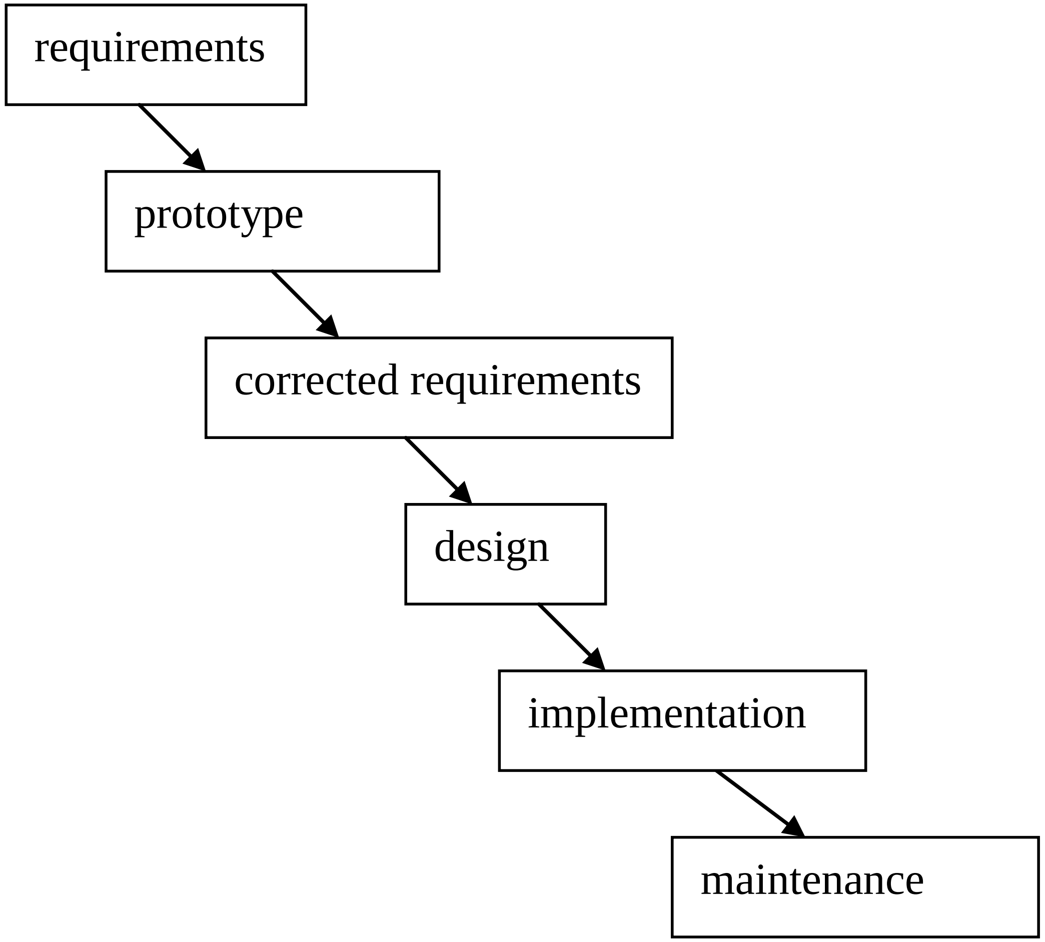
Lifecycle vs. lifespan model

- Lifecycle
  - common terminology
  - Rajlich argues that the term is incorrect: there is no cycle
    - some software discontinued without a replacement
- Lifespan model
  - Rajlich argues it is a better terminology
  - less commonly used

V-Model


maintenance
requirements
functional testing
system design
system testing
unit design
unit testing
implementation

Prototyping model


requirements
prototype
corrected requirements
design
implementation
maintenance