

Towards a Quality Model for Open Source Software

Siraj Shaikh and Antonio Cerone

United Nations University

International Institute for Software Technology

Macau SAR China

www.iist.unu.edu/~siraj, www@iist.unu.edu/~antonio

Outline

- Motivations

- Views about quality
- Empirical Analysis
- Weaknesses

in the Open Source Methodology

- Notions of Quality for OSS

- Factors providing quality
- Relationships among factors

- Discussion

- Towards a Quality Metrics for OSS
- Where do we go from here?

Raymond's View

The *high level* of quality of free software is partly due to the high degree of peer review and user involvement [Raymond 1999]

Raymond's View

The *high level* of quality of free software is partly due to the high degree of peer review and user involvement [Raymond 1999]

This hypothesis

- is based on anecdotal rather than empirical evidence

Raymond's View

The *high level* of quality of free software is partly due to the high degree of peer review and user involvement [Raymond 1999]

This hypothesis

- is based on anecdotal rather than empirical evidence
- needs to be validated and tested rigorously through empirical data analysis

Open Source Methodology

based on:

Open Source Methodology

based on:

- frequent beta release

Open Source Methodology

based on:

- frequent beta release
- decentralised contribution to a code base

Open Source Methodology

based on:

- frequent beta release
- decentralised contribution to a code base
- accessibility as open as possible

Open Source Methodology

based on:

- frequent beta release
- decentralised contribution to a code base
- accessibility as open as possible
- extensive peer review

Open Source Methodology

based on:

- frequent beta release
- decentralised contribution to a code base
- accessibility as open as possible
- extensive peer review
- ignoring the traditional software models
[O'Reilly 1999] [Raymond 1999]

Open Source Methodology

based on:

- frequent beta release
 - decentralised contribution to a code base
 - accessibility as open as possible
 - extensive peer review
 - ignoring the traditional software models
[O'Reilly 1999] [Raymond 1999]
- ⇒ yields products in rapid succession and often with high quality

Open Source Methodology

based on:

- frequent beta release
- decentralised contribution to a code base
- accessibility as open as possible
- extensive peer review
- ignoring the traditional software models
[O'Reilly 1999] [Raymond 1999]

⇒ yields products in rapid succession and often with high quality

⇒ may distort common approaches to quality assurance

Beta vs. Stable Realeases

- frequent **beta releases**

Beta vs. Stable Realeases

- frequent **beta releases** contain
 - bug fixes
 - new features **lightly tested**

Beta vs. Stable Realeases

- frequent **beta releases** contain
 - bug fixes
 - new features **lightly tested**

typical frequency: every 2–3 weeks

Beta vs. Stable Realeases

- frequent **beta releases** contain
 - bug fixes
 - new features **lightly tested**

typical frequency: every 2–3 weeks
- periodic **stable releases**

Beta vs. Stable Realeases

- frequent **beta releases** contain
 - bug fixes
 - new features **lightly tested**

typical frequency: every 2–3 weeks
- periodic **stable releases**
 - supposedly **more extensively tested**

Beta vs. Stable Realeases

- frequent **beta releases** contain
 - bug fixes
 - new features **lightly tested**

typical frequency: every 2–3 weeks
- periodic **stable releases**
 - supposedly **more extensively tested**

typical frequency: every year

[Yilmaz et al. 2006]

Key Elements of the OpenSource Methodology

Key Elements

of the OpenSource Methodology

- frequent beta release
- decentralised contribution to a code base
- accessibility as open as possible
- extensive peer review
- ignoring the traditional software models

Decentralised Contribution

Who are the contributors to code development?

Decentralised Contribution

Who are the contributors to code development?

- unpaid volunteers

Decentralised Contribution

Who are the contributors to code development?

- unpaid volunteers
- who work on whatever they want

Decentralised Contribution

Who are the contributors to code development?

- unpaid volunteers
- who work on whatever they want
- may have a wide range of expertise

Decentralised Contribution

Who are the contributors to code development?

- unpaid volunteers
- who work on whatever they want
- may have a wide range of expertise
- have variable levels of skills

Decentralised Contribution

Who are the contributors to code development?

- unpaid volunteers
- who work on whatever they want
- may have a wide range of expertise
- have variable levels of skills
- have different approaches to development

Decentralised Contribution

Who are the contributors to code development?

- unpaid volunteers
- who work on whatever they want
- may have a wide range of expertise
- have variable levels of skills
- have different approaches to development
- have a volatile nature

Decentralised Contribution

Who are the contributors to code development?

- unpaid volunteers
- who work on whatever they want
- may have a wide range of expertise
- have variable levels of skills
- have different approaches to development
- have a volatile nature

[Michlmyer et al. 2005]

Code-level Peer Review

Initial Release

- kernel of code
- made available on the Internet
- for review

Code-level Peer Review

Initial Release

- kernel of code
- made available on the Internet
- for review

Then

- **screen changes** to code base
- when code base too big, **delegate responsibility for major components**

Efficient Review Process?

- **large number** of reviewers contributes to bug fixes

Efficient Review Process?

- **large number** of reviewers contributes to bug fixes
- **fast** bug correction

Efficient Review Process?

- **large number** of reviewers contributes to bug fixes
- **fast** bug correction
- \implies **effective** development (**visible result**)

Efficient Review Process?

- **large number** of reviewers contributes to bug fixes
- **fast** bug correction
- \implies **effective** development (**visible result**)

large number \wedge fast \wedge effective \neq efficient

Efficient Review Process?

- **large number** of reviewers contributes to bug fixes
- **fast** bug correction
- \implies **effective** development (**visible result**)

large number \wedge fast \wedge effective \neq efficient

fast \wedge effective = **seen result of the effort**

Efficient Review Process?

- **large number** of reviewers contributes to bug fixes
- **fast** bug correction
- \implies **effective** development (**visible result**)

large number \wedge **fast** \wedge **effective** \neq **efficient**

fast \wedge **effective** = **seen result of the effort**

large number = **unseen magnitude of the effort**

Efficient Review Process?

- **large number** of reviewers contributes to bug fixes
- **fast** bug correction
- \implies **effective** development (**visible result**)

large number \wedge **fast** \wedge **effective** \neq **efficient**

fast \wedge **effective** = **seen result of the effort**

large number = **unseen magnitude of the effort**

Question: “Does this methodology reduces development costs overall, or does it just push effort into dark economic corners where it is harder to see?” [McConnell 1999]

What to Do

Fix **four major problems** [McConnell 1999]:

- create central **clearinghouse** for the open-source methodology
- overcome its **addiction to code and fix**
- eliminate **upstream defects earlier**
- collect and publish data to support **effectiveness** of the open-source methodology

What has been Done

Fix **four major problems** [McConnell 1999]:

- create central **clearinghouse** for the open-source methodology
 - ⇒ **proposals** [Halloran and Scherlis 2002] [Yilmaz et al. 2006]
- overcome its **addiction to code and fix**
- eliminate **upstream defects earlier**
- collect and publish data to support **effectiveness** of the open-source methodology
 - ⇒ **surveys** [Zhao and Elbaum 2000 + 2003]
 - ⇒ **interviews** [Michlmayr 2005] [Michlmayr et al. 2005]
 - ⇒ **case studies** [Mockus et al. 2002]

Anything More?

Fix **four major problems** [McConnell 1999]:

- create central **clearinghouse** for the open-source methodology
- overcome its **addiction to code and fix**
- eliminate **upstream defects earlier**
- collect and publish data to support **effectiveness** of the open-source methodology

Anything More?

Fix **four major problems** [McConnell 1999]:

- create central **clearinghouse** for the open-source methodology
- overcome its **addiction to code and fix**
- eliminate **upstream defects earlier**
- collect and publish data to support **effectiveness** of the open-source methodology

through

- notions of **quality for OSS** to provide a **formal framework**

Traditional Notions of Quality

[IEEE 1999]

the degree to which a system, component, or process meets (1) specified requirements, (2) customers or user needs or expectations

Traditional Notions of Quality

[IEEE 1999]

the degree to which a system, component, or process meets (1) specified requirements, (2) customers or user needs or expectations

[Pressman 2000]

conformance to explicitly stated functional and performance requirements, explicitly documented developments standards, and implicit characteristics that are expected of all professionally developed software

Quality for OSS

- notions of quality for OSS

Quality for OSS

- notions of **quality** for OSS
- understand the **factors** contributing to quality

Quality for OSS

- notions of **quality for OSS**
- understand the **factors** contributing to quality
- discover **relationships** among factors

Positive Perceptions

[Michlmyer et al. 2005] [Yilmaz et al. 2006]:
Perception of a **potential quality** due to

Positive Perceptions

[Michlmyer et al. 2005] [Yilmaz et al. 2006]:

Perception of a **potential quality** due to

- **intensive feedback** promoted by the open development may improve the software

Positive Perceptions

[Michlmyer et al. 2005] [Yilmaz et al. 2006]:

Perception of a **potential quality** due to

- **intensive feedback** promoted by the open development may improve the software
- **high motivation** due to the fact that volunteers can work on whatever they want

Positive Perceptions

[Michlmyer et al. 2005] [Yilmaz et al. 2006]:

Perception of a **potential quality** due to

- **intensive feedback** promoted by the open development may improve the software
- **high motivation** due to the fact that volunteers can work on whatever they want
- **better human resources** due to the wide range of expertise and knowledge of developers

Negative Perceptions

[Michlmyer et al. 2005] [Yilmaz et al. 2006]:
Perception of a **several problems** due to

Negative Perceptions

[Michlmyer et al. 2005] [Yilmaz et al. 2006]:

Perception of a **several problems** due to

- **unmaintained and unsupported code** due to the volatile nature of volunteers

Negative Perceptions

[Michlmyer et al. 2005] [Yilmaz et al. 2006]:

Perception of a **several problems** due to

- **unmaintained and unsupported code** due to the volatile nature of volunteers
- **useless bug reports** due to user with few technical skills

Negative Perceptions

[Michlmyer et al. 2005] [Yilmaz et al. 2006]:

Perception of a **several problems** due to

- **unmaintained and unsupported code** due to the volatile nature of volunteers
- **useless bug reports** due to user with few technical skills
- **lack of testing** because contributors are more interested in developing code

Negative Perceptions

[Michlmyer et al. 2005] [Yilmaz et al. 2006]:

Perception of a **several problems** due to

- **unmaintained and unsupported code** due to the volatile nature of volunteers
- **useless bug reports** due to user with few technical skills
- **lack of testing** because contributors are more interested in developing code
- **lack of documentation** about already developed code and development practices

Negative Perceptions

[Michlmyer et al. 2005] [Yilmaz et al. 2006]:

Perception of a **several problems** due to

- **unmaintained and unsupported code** due to the volatile nature of volunteers
- **useless bug reports** due to user with few technical skills
- **lack of testing** because contributors are more interested in developing code
- **lack of documentation** about already developed code and development practices
- **poor coordination and communication** due to unclear responsibilities

Factors Providing Quality

Open Source Methodology based on:

- frequent beta release
- decentralised contribution to a code base
- accessibility as open as possible
- extensive peer review
- ignoring the traditional software models

Extracting Factors

Factors:

- frequent beta release
 - decentralised contribution to a code base
- accessibility as open as possible
 - extensive peer review
- ignoring the traditional software models

Extracting Factors

Factors:

- frequent beta release
 - decentralised contribution to a code base
- accessibility as open as possible
 - extensive peer review
- ignoring the traditional software models

How do such factors affect quality?

Extracting Factors

Factors:

- frequent beta release
 - decentralised contribution to a code base
- accessibility as open as possible
 - extensive peer review
- ignoring the traditional software models

How do such factors affect quality?

⇒ need quality measure

Quality Measures

Quality by

- **development: frequent beta release**
 - decentralised contribution to a code base
- **access: accessibility as open as possible**
 - extensive peer review
- **design: ignoring the traditional software models**

Relationships among Factors



quality provided by **Factor 2** is dependent on
on quality provided by **Factor 1**

Relationships among Factors



quality provided by **Factor 2** is dependent on
on quality provided by **Factor 1**



Factor 1 and **Factor 2** complement each other

Quality by Development

frequent beta release

decentralised contribution to a code base

Quality by Development

decentralised contribution to a code base

Quality by Development

decentralised contribution to a code base

effective communication
coordination and management

[Halloran and Scherlis 2002]

communication: mediated through project server

⇒ **project wall**

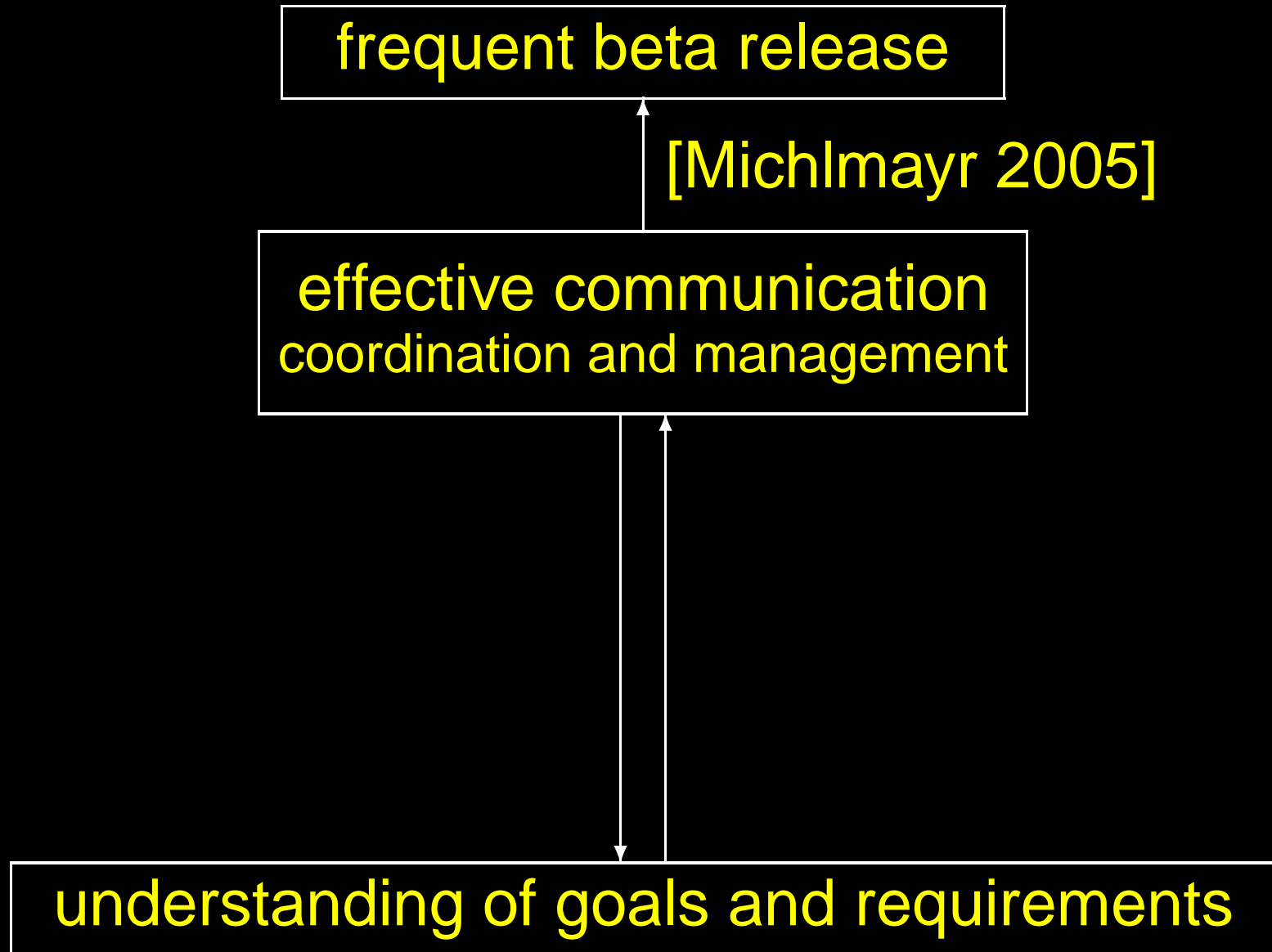
coordination: responsibility, privileges, releases

management: adoption of only open source tools

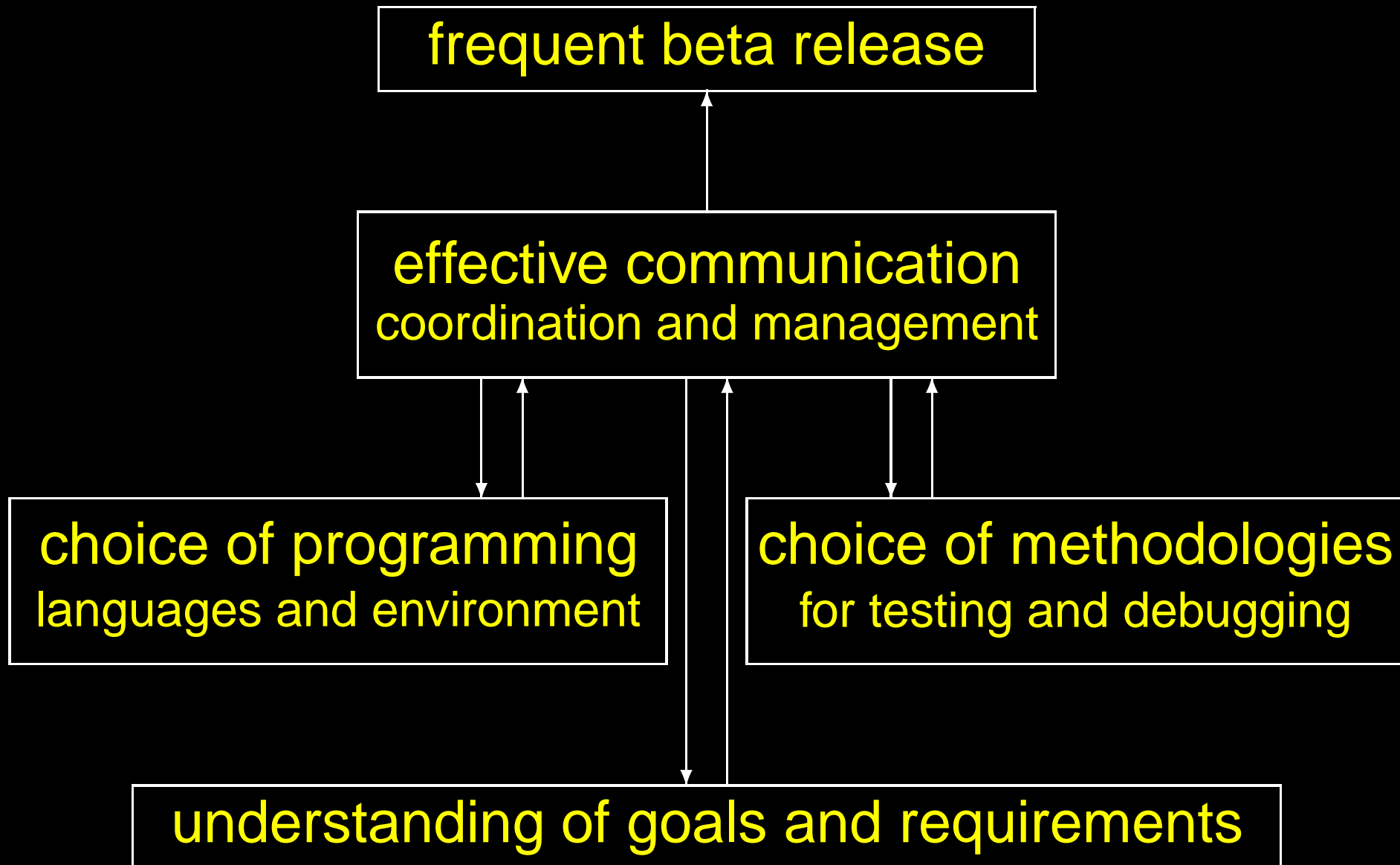
⇒ “gentle slope” learning curve

⇒ **ideology** or **effective practice?**

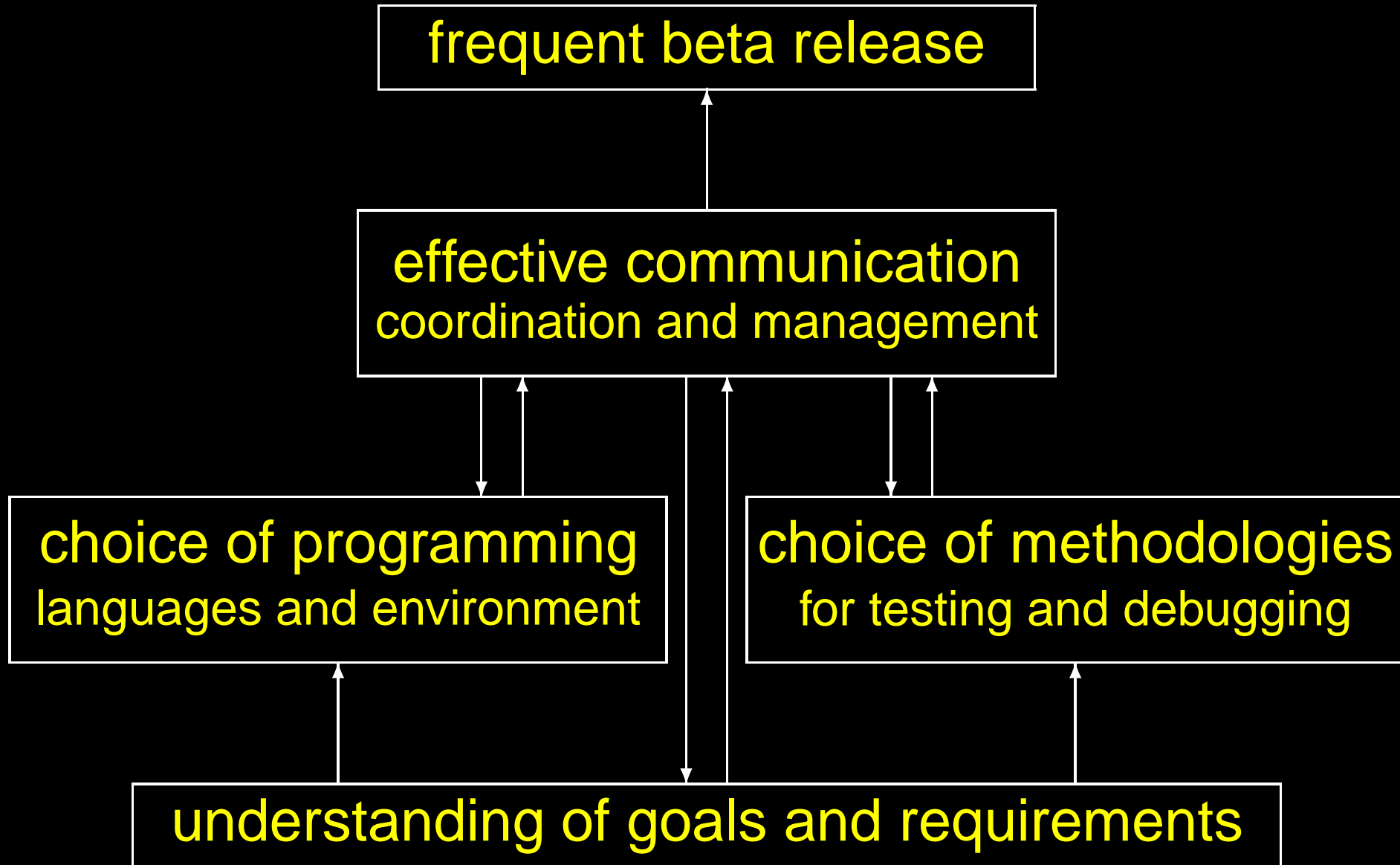
Quality by Development



Quality by Development



Quality by Development



Quality by Access

specific for OSS

Quality by Access

specific for OSS

suitable
format

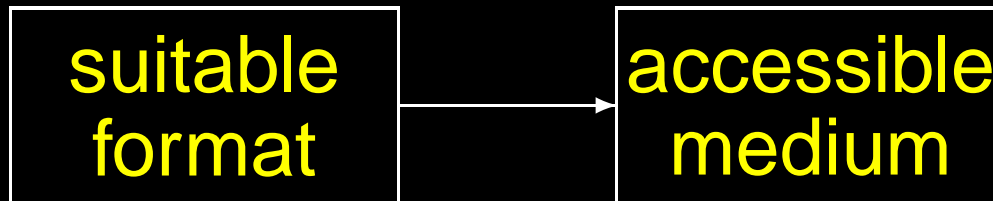
attractive for
developers

workable for
development and review

appropriate for
free distribution

Quality by Access

specific for OSS



the Internet
by means of
project web server
bug tracking systems
version control systems
mailing lists
discussion forum

Quality by Access

specific for OSS



project wall:
maximise outgoing
information flow

Quality by Access

specific for OSS



Design Techniques

use of recognised design
and engineering techniques

Design Techniques

use of recognised design
and engineering techniques

Anyone is allowed to view and copy the source code

Design Techniques

use of recognised design
and engineering techniques

Anyone is allowed to view and copy the source code
What do the project leaders actually control?

Design Techniques

use of recognised design
and engineering techniques

Anyone is allowed to view and copy the source code

What do the project leaders actually control?

Control composition, configuration, information flow

Design Techniques

use of recognised design
and engineering techniques

Anyone is allowed to view and copy the source code
What do the project leaders actually control?

Control composition, configuration, information flow
⇒ tight control over the engineering practices

not by limiting

individual developers in their personal space

but by limiting

the kind of transaction they can make
with the persistent project state

[Halloran and Scherlis 2002]

Documentation

use of recognised design
and engineering techniques



```
graph LR; A[use of recognised design and engineering techniques] --> B[frequently updated documentation]
```

frequently updated
documentation

Documentation

use of recognised design
and engineering techniques



frequently updated
documentation

about

already developed code

development practice

- * code styles
- * code comit

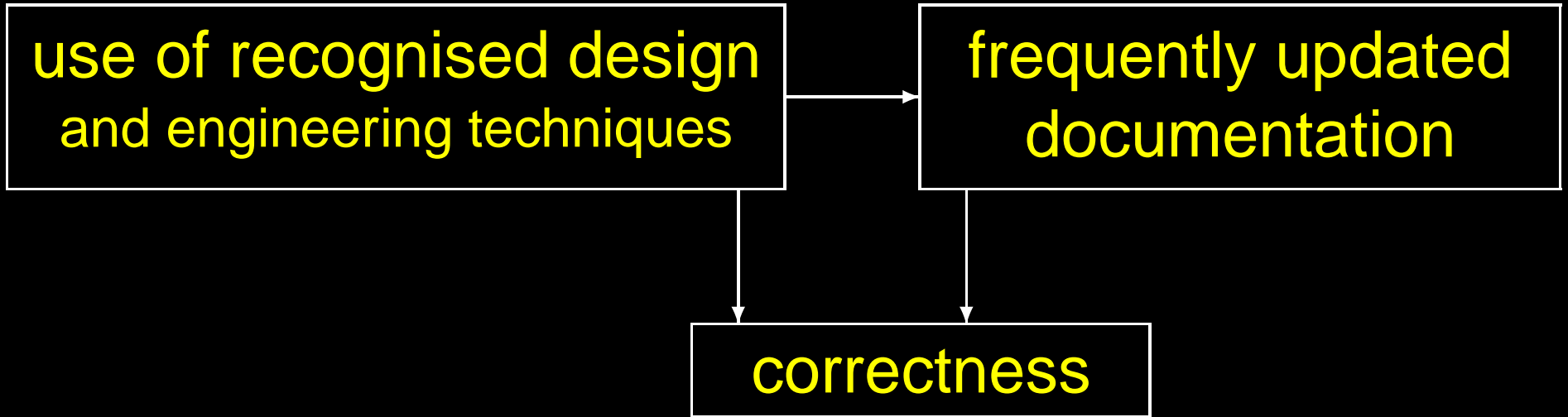
Quality by Design

use of recognised design
and engineering techniques

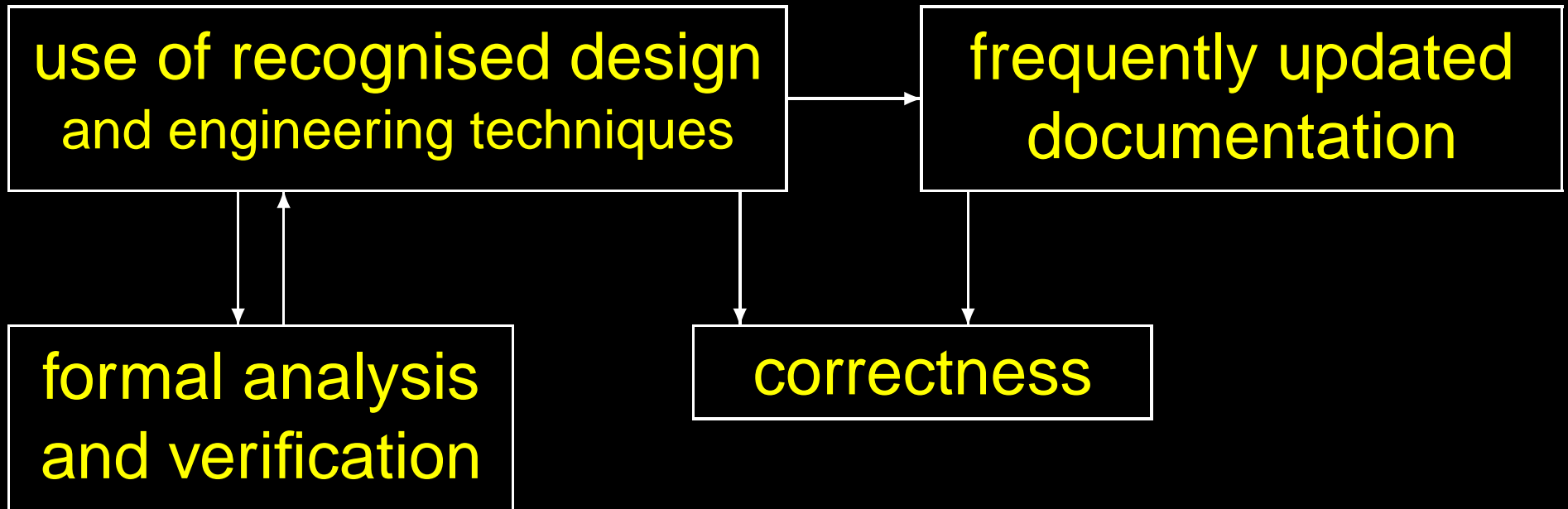


frequently updated
documentation

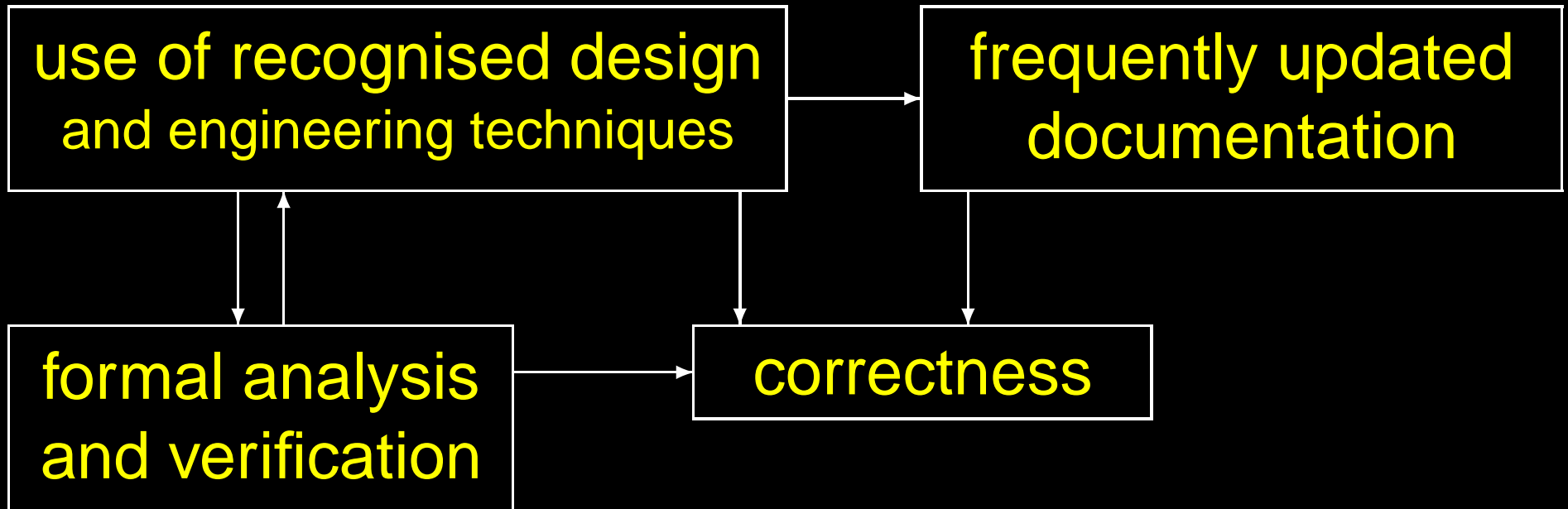
Quality by Design



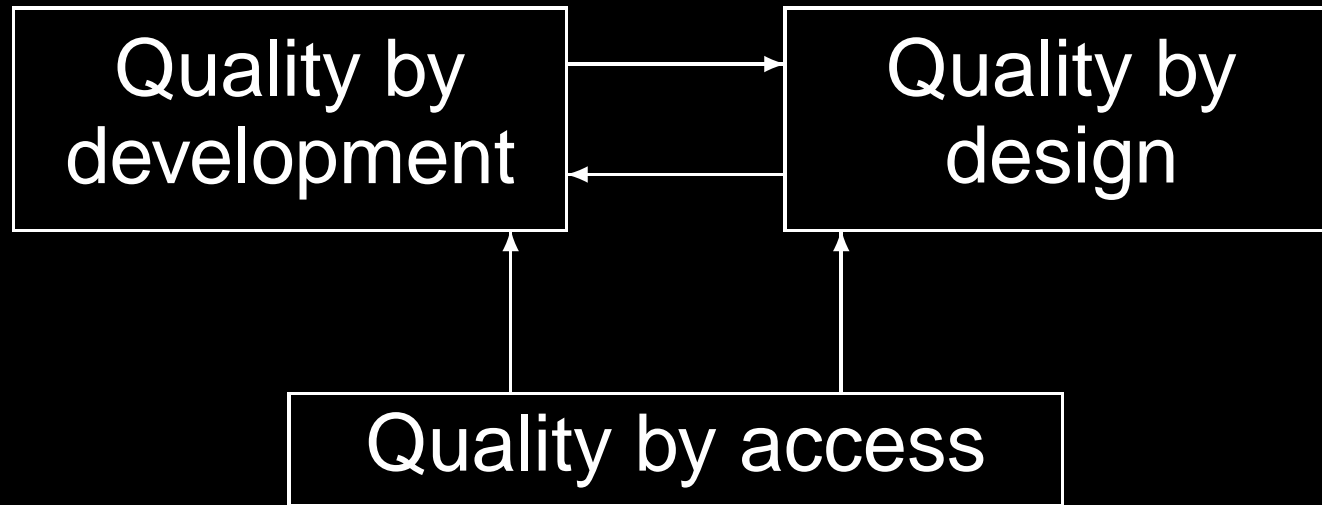
Quality by Design



Quality by Design



Quality Measure Relationships



Towards a Quality Metrics

- relationships among quality factors aim to define a **framework** on which to build a **quality metrics** for OSS

Towards a Quality Metrics

- relationships among quality factors aim to define a **framework** on which to build a **quality metrics for OSS**
- **factors and relationships** need to be
 - further **refined**
 - made **quantitative**

Towards a Quality Metrics

- relationships among quality factors aim to define a **framework** on which to build a **quality metrics for OSS**
- **factors** and **relationships** need to be
 - further **refined**
 - made **quantitative**
 - **quantitative measures** for factors
 - **varying degrees of dependency** for relationships

Quality Metrics contributes to

Fix **four major problems** [McConnell 1999]:

- create central **clearinghouse** for the open-source methodology
- overcome its **addiction to code and fix**
- eliminate **upstream defects earlier**
- collect and publish data to support **effectiveness** of the open-source methodology

What has been Done

Fix **four major problems** [McConnell 1999]:

- create central **clearinghouse** for the open-source methodology
 - ⇒ **proposals** [Halloran and Scherlis 2002] [Yilmaz et al. 2006]
- overcome its **addiction to code and fix**
- eliminate **upstream defects earlier**
- collect and publish data to support **effectiveness** of the open-source methodology
 - ⇒ **surveys** [Zhao and Elbaum 2000 + 2003]
 - ⇒ **interviews** [Michlmayr 2005] [Michlmayr et al. 2005]
 - ⇒ **case studies** [Mockus et al. 2002]

What's Next

Fix four major problems [McConnell 1999]:

- create central **clearinghouse** for the open-source methodology **through better** Quality by development
- overcome its **addiction to code and fix** **through better** Quality by design
- eliminate **upstream defects earlier** **through better** Quality by design
- collect and publish data to support **effectiveness** of the open-source methodology **and improve** Quality by access

OpenCert Community

- **Where** do we go from here?

OpenCert Community

- **Where** do we go from here?
- Analyse a **common case study**?

OpenCert Community

- **Where** do we go from here?
- Analyse a **common case study**?
- Collect **more data**?
(e.g. with interviews and questionnaires)

OpenCert Community

- **Where** do we go from here?
- Analyse a **common case study**?
- Collect **more data**?
(e.g. with interviews and questionnaires)
- Develop together a **common case study**?

OpenCert Community

- **Where** do we go from here?
- Analyse a **common case study**?
- Collect **more data**?
(e.g. with interviews and questionnaires)
- Develop together a **common case study**?
 - agree on **modelling and design methodologies and tools**

OpenCert Community

- **Where** do we go from here?
- Analyse a **common case study**?
- Collect **more data**?
(e.g. with interviews and questionnaires)
- Develop together a **common case study**?
 - agree on **modelling and design methodologies and tools**
 - produce/identify a **model** and a **small kernel of code** and make it **available on the Internet**

OpenCert Community

- **Where** do we go from here?
- Analyse a **common case study**?
- Collect **more data**?
(e.g. with interviews and questionnaires)
- Develop together a **common case study**?
 - agree on **modelling and design methodologies and tools**
 - produce/identify a **model** and a **small kernel of code** and make it **available on the Internet**
 - incrementally introduce methods and tools

References

General References

- [O'Reilly 1999]
IEEE STD 610.12–1990
Lessons From Open-Source Development
Prentice Hall, 1999
- [Raymond 1999]
E. S. Raymond
The Cathedral and the Bazaar
O'Reilly and Associates, 1999
- [McConnell 1999]
S. McConnell
Open Source Methodology: Ready for prime time?
IEEE Software, 16(4):6–8, 1999, IEEE Computer Society, 1999

References: Quality

- [IEEE 1999]
IEEE STD 610.12–1990
IEEE Standard Glossary of Software Engineering Terminology
- [Pressman 2000]
S. R. Pressman
Software Engineering — A Practitioner's Approach
McGraw-Hill International, 2000

References: Reports on Surveys

- [Zhao and Elbaum 2000]

L. Zhao and S. Elbaum

A survey on quality related activities in open source

ACM SIGSOFT Engineering Notes, 25(3):53–57, 2000,

ACM Press

- [Zhao and Elbaum 2003]

L. Zhao and S. Elbaum

Quality assurance under the open source development model

Journal of System and Software, 66(1):65–75, 2003,

Elsevier Science

References: Proposals (from Empirical Studies)

- [Halloran and Scherlis 2002]

T. J. Halloran and W. L. Scherlis

High quality and open source software practices

Proc. of 2nd Workshop on Open Source Software Engineering, 2002

- [Yilmaz et al. 2006]

C. Yilmaz, A. M. Memon, A. Porter, A. S. Krisna, D. C. Schmidt, A. Gokhale

Techniques and processes for improving the quality and performance of open-source software

2006

References: Interview-based Analyses

- [Michlmayr 2005]

M. Michlmayr

Quality improvement in volunteer free software projects: Exploring the impact of release management
Proc. of 1st Int. Conf. on Open Source Systems, pages 309-310, Genova, Italy, 2005

- [Michlmayr et al. 2005]

M. Michlmayr, F. Hunt, D. Probert

Quality practises and problems in free software projects
Journal of System and Software, 66(1):65–75, 2003,
Elsevier Science
Proc. of 1st Int. Conf. on Open Source Systems, pages 24-28, Genova, Italy, 2005

References: Case Studies

- [Mockus et al.]

A. Mockus, R. T. Fielding, J. D. Herbsleb

Two case studies of open source development: Apache and Mozilla

ACM Transactions on Software Engineering and Methodology, 11(3):309–346, 2002, ACM Press