

CSMR'2001
5th European Conference on
Software Maintenance and Reengineering
Lisbon, March 14, 2001

Is there something like light maintenance?
Maintenance in the light of light
methodologies

Karol Frühauf, *INFOGEM AG*, CH-5401 Baden, Switzerland
Karol.Fruehauf@ACM.ORG

Content

1. heavy and light methodologies
2. suitability of methodologies for different software categories
3. the nature of maintenance
4. maintenance in the light world
5. conclusions

Heavy and light methodologies

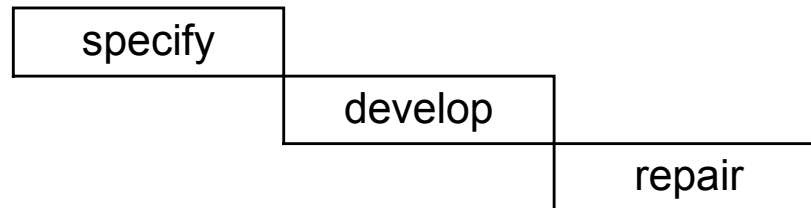
methodology = how do we do things

type	name	tag	hints
heavy	CMM	US DoD	procurement
	ISO 9001	world-wide	procurement in general
	ISO 12207	world-wide, SPICE	procurement of software and related services
	V-Modell	D, administration	procurement
	RUP	Rational proprietary	supplier oriented
light	Extreme Programming	Kent Beck	
	Lean Development	Bob Charette	
	Crystal methods	Alistair Cockburn	
	Adaptive Software Development	Jim Highsmith	
	Scrum	Ken Schwaber	

Characteristics of evolutionary software development

with light or heavy methodology

release i
development
project



product
life cycle



customer X(i)
project



Basic difference in the mental models

heavy methodologies

project execution can be standardised

customer involvement is unlikely
requirements need to be defined to a large extent up front

architecture need to be fixed before design and coding starts
it takes time to make something useful for the customer

“do it right the first time” is the right thing to do

light methodologies

no two projects will ever be the same

customer involvement is critical
a feeling for the whole is needed up front, but only requirements for the current feature can be clearly defined

design for the current feature, not for the future

customer value must be delivered every six weeks

try, try again

Ingredients of the methodologies

heavy

- process model and descriptions
- project plan templates
- process result templates

light

- principles
- practices
- people skills

Light versus heavy methodologies

topic	principle	
	light methodologies	heavy methodologies
knowledge	tacit, in people	explicit, process
communication	person-to-person	via documentation
collaboration	continuous	more or less frequent
attitude to change	change tolerant	change resistant
attitude to risk	adapting to risk	anticipating risk
practices	build up from absolute minimum	tailor down from maximum
discipline	high, informal	formal, low

Categories of software products

depending on the application area

embedded software	embedded in hardware
information systems	interactive, data base applications
system software	makes hardware use easier
applications	stand-alone software packages
e-* systems	more than displaying a home page

depending on the supplier / customer relationship

made for house	internal customer, internal problem
individual	external customer, external problem
made for market	internal customer, external problem

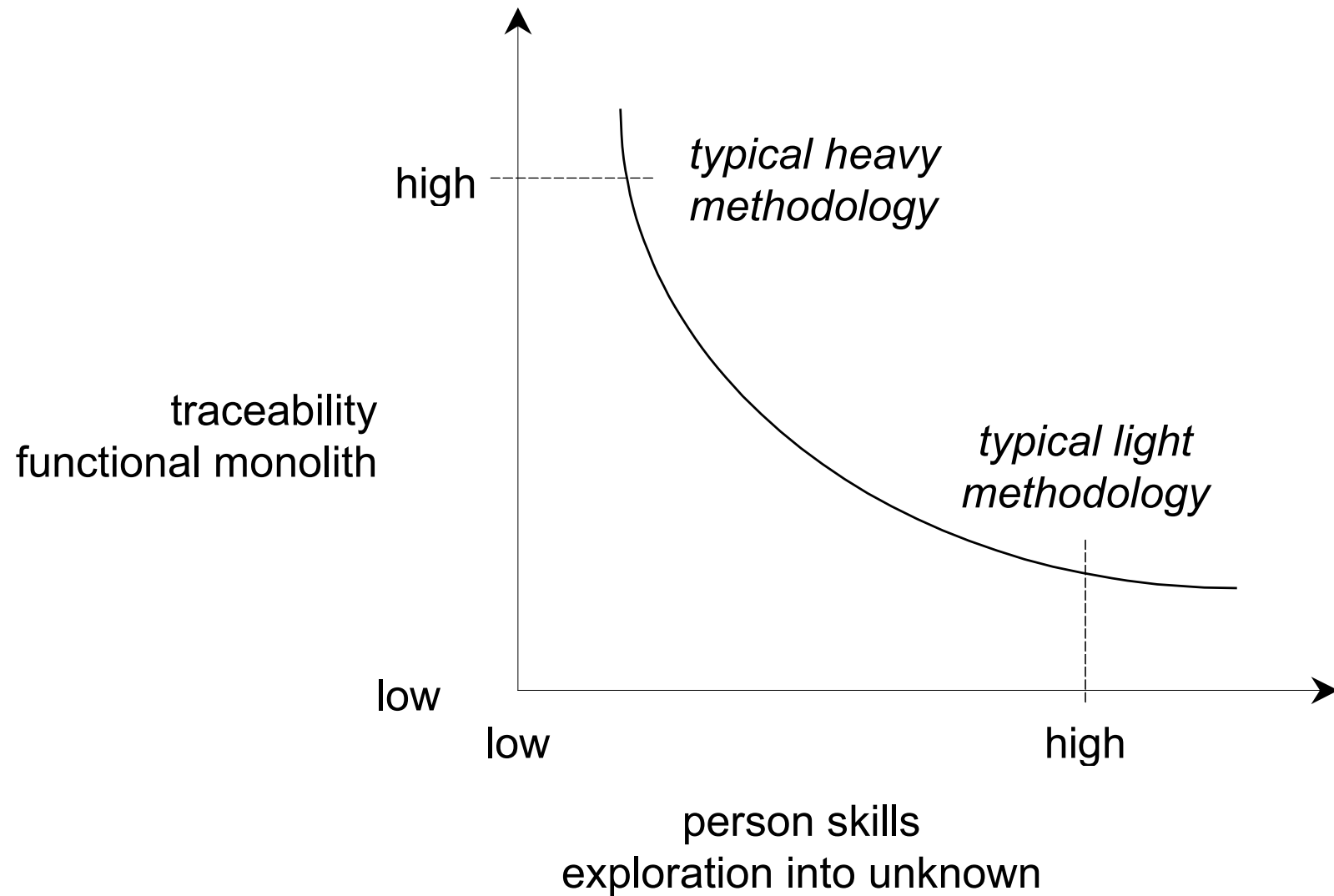
Suitability of methodologies for software categories

	made for house	individual	made for market
embedded software	heavy	heavy	heavy
information systems	light	<i>light .. heavy</i>	heavy
system software	–	–	heavy
applications	light	<i>light .. heavy</i>	<i>light .. heavy</i>
e-* systems	light	<i>light .. heavy</i>	<i>light .. heavy</i>

Key characteristics

1. traceability
high requirements on safety, security, reliability
2. frequency of delivery
customer / production effort needed at each delivery for
 - installation
 - data migration
 - training
 - ⇒ frequent „delivery” for integration, not production
 - ⇒ periodic delivery to the customer / into production
3. frequency of changes, degree of problem understanding
business innovation, rapid technology development

Influence of product and project characteristics



Development, maintenance, repair

to develop to cause to grow or expand

to maintain to cause to remain unaltered or unimpaired

to repair to make (something) good, strong, whole etc. after damage, injury etc.

aim of hardware maintenance

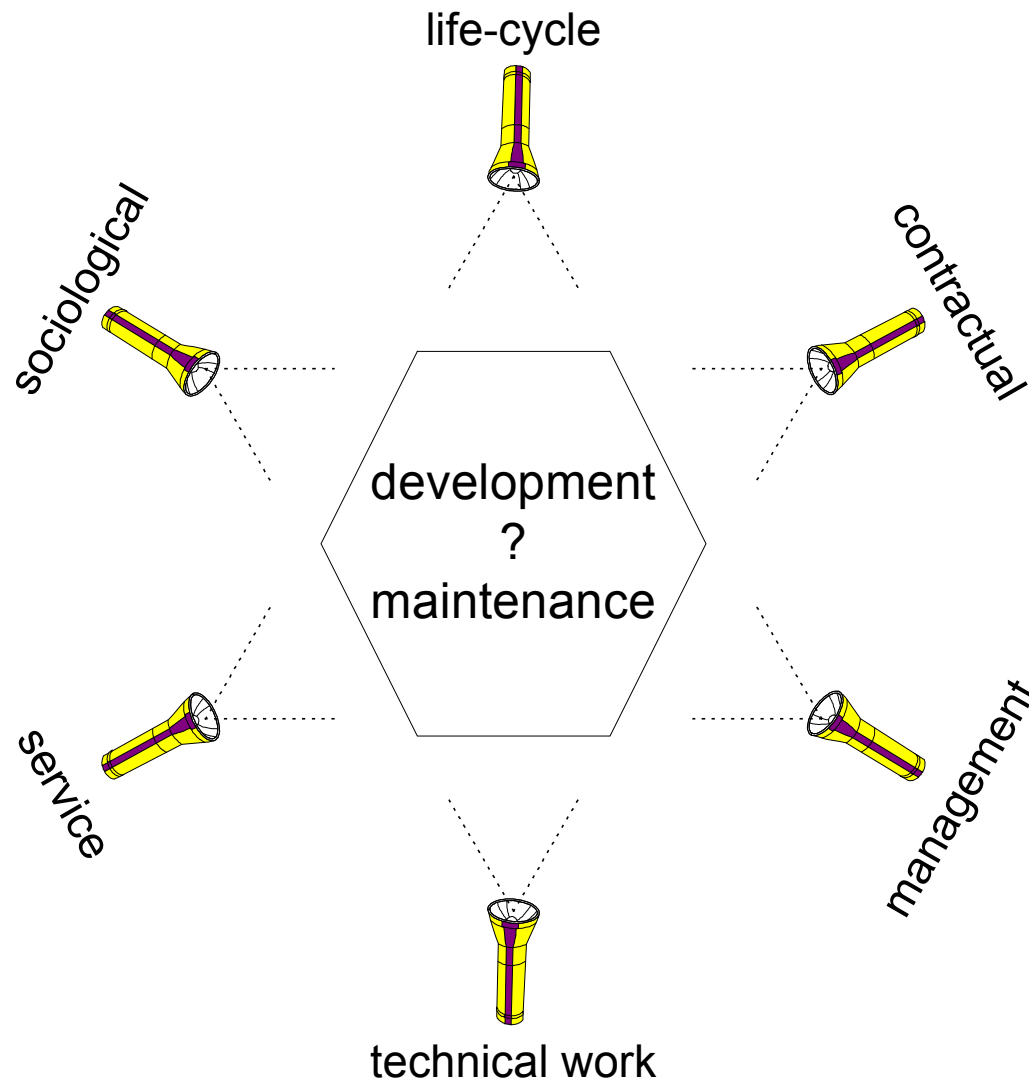
to cause the hardware to *regain* the characteristics *it had* at the start of its utilisation

aim of software maintenance

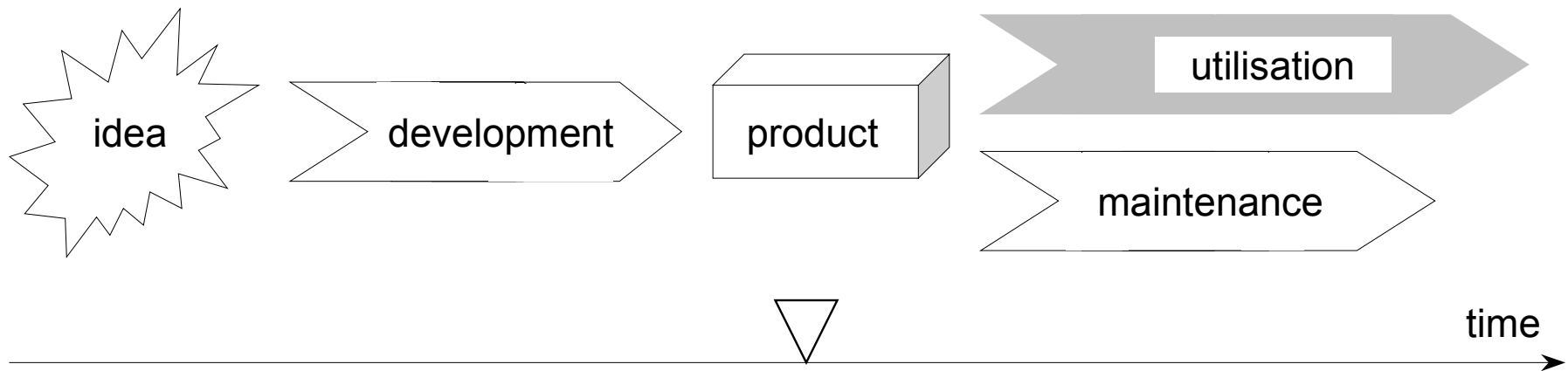
to cause the software to *gain* the characteristics it *should have had* at the start of its utilisation

→ repair after “damage, injury“ *during* development

Aspects of development and maintenance

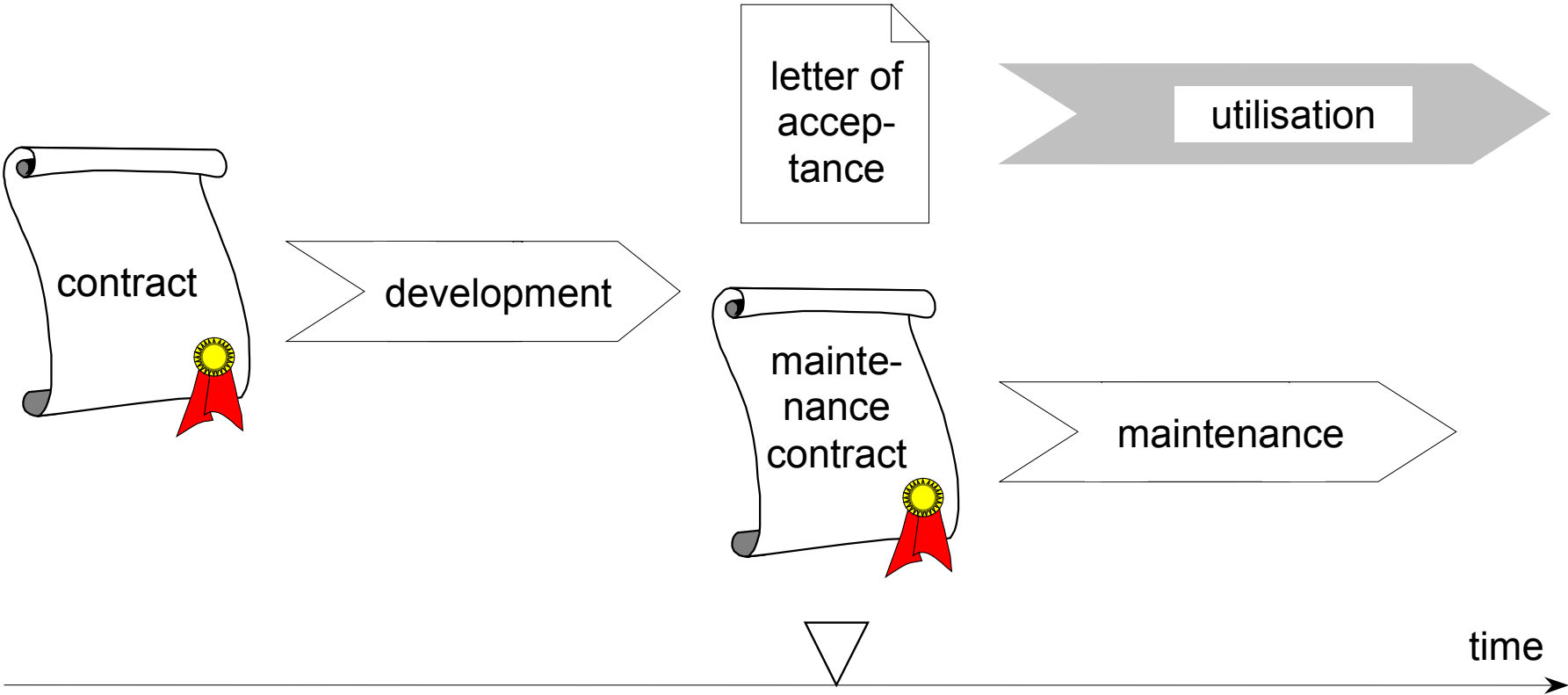


Periods in the product life cycle



first release?
second release?
:
last release?

Periods covered by contracts



Management aspects (supplier side)

topic	development	maintenance
organisation	project	line organisation and / or project
scope of the work	whole product	parts of the product
pieces of work	rather big	rather small
type of work	planned	event driven & planned
reliable planning	possible	very unlikely
controlling	work packages, milestones	problem reports, change requests

⇒ management approaches differ

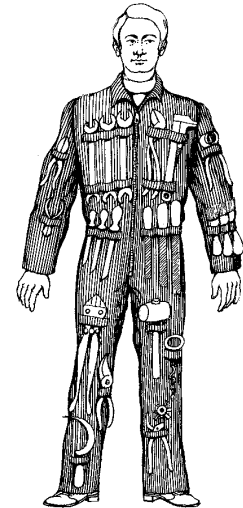
Technical work content

topic	development	maintenance
requirements	not stable	fairly stable and clear
design	evolving	is a restriction
coding	new (and reused) code	existing code
reviewing	on all levels of abstraction	mainly code, may be design changes
testing	module, integration, system tests	change test, regression test
configuration management	necessary	essential
data migration	occasionally	regularly

⇒ main difference in the activities (re)designing, (re)coding and (re)testing

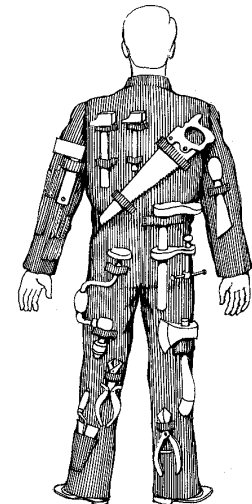
Service aspects

during development	no service to the external customer
end of development	training and consulting
maintenance	hot-line with first level support second level support disaster recovery (help) consulting training



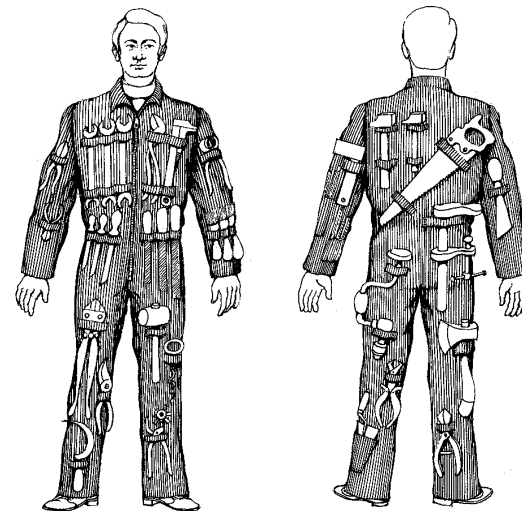
Sociological aspects

- development is a highly estimated job
creative co-operation in teams
requires (managerial and technical)
leadership
is fun
- maintenance is considered to be a job of less value
deductive work alone
requires (managerial) co-ordination
and mutual (technical) consulting
is less fun but very instructive



Characteristics of maintenance – summary

1. begins with the start of the product use
2. is covered by a separate contract
3. management by quick decisions
4. restrictions by the existing design and employed technology
5. major service activities
6. a challenge but not much fun



Maintenance within light methodologies

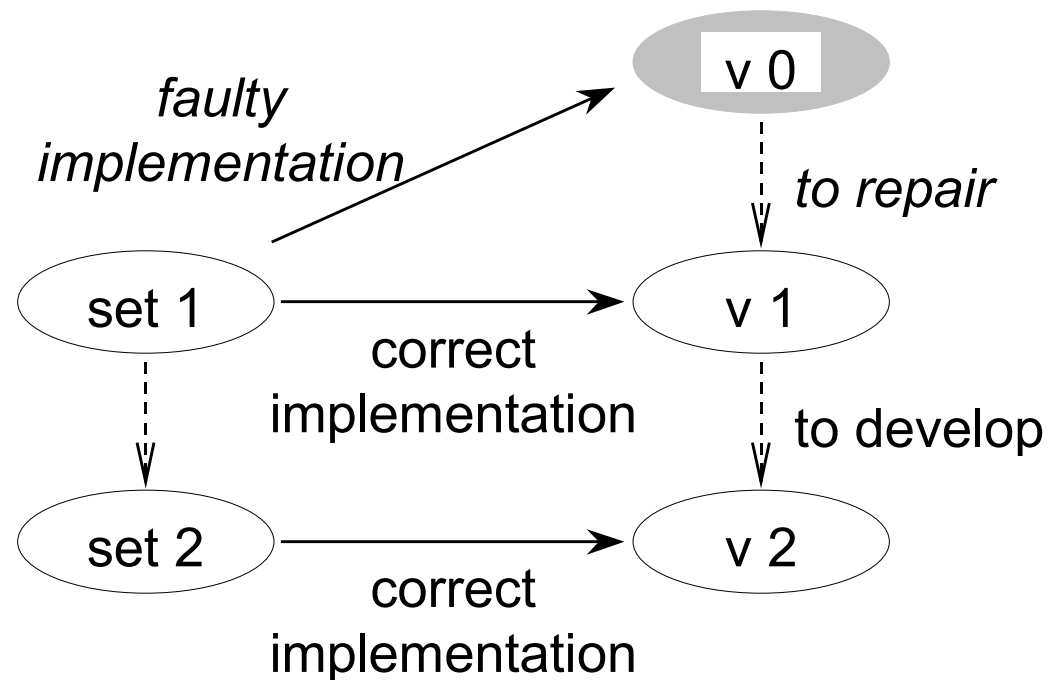
1. begins with the start of the product use
2. contractual framework and negotiations for each feature
3. management by communication and collaboration
4. the existing design and technology is considered to be a “best trial”, their modification is inherent (not an exception)
5. no big difference in service activities
6. the rules of the game ensure that the eternal change is not only a challenge but also fun

Conclusions for software maintenance

to maintain to develop (enhance, perfect, improve) &
 to repair (correct, fix, remove defects)

requirements

software



while the software product is in use

Is there something like light maintenance?

the answer is *yes*

- light methodologies = maintenance oriented methodologies concerning technical work and its management
the transition from “development” to “maintenance” is smooth

the answer is *no*

- even with light methodologies maintenance is not a light job to do
concerning customer services and their management
delivery, customisation, installation, putting into production is not covered by the light methodologies

Literature

Beck (2000)

Beck, Kent: **Extreme Programming Explained – Embrace Change.**

Addison-Wesley, 2000, ISBN 0-201-61641-6

Frühau, Jeppesen (1987)

Frühau, Karol; Jeppesen, Klaus Jul: The staircase approach.

Proceedings of the IFAC/IFIP Workshop on Experience with the Management of Software Projects, Heidelberg, May 1986, Pergamon Press, 1987, ISBN 0-08-034089-X

Highsmith (2000)

Highsmith, Jim: E-Project Management: Harnessing Innovation and Speed.

Cutter Consortium, Executive Report, Vol. 1, No. 1, 2000

Ludewig (2000)

Ludewig, Jochen: **Lectures on Software Engineering.**

University Stuttgart, 2000.

Pigoski (1996)

Pigoski, Thomas M: **Practical Software Maintenance.**

John Wiley, 1996, ISBN 0-471-17001-1