"Software Patents" - IT-Security at Stake?

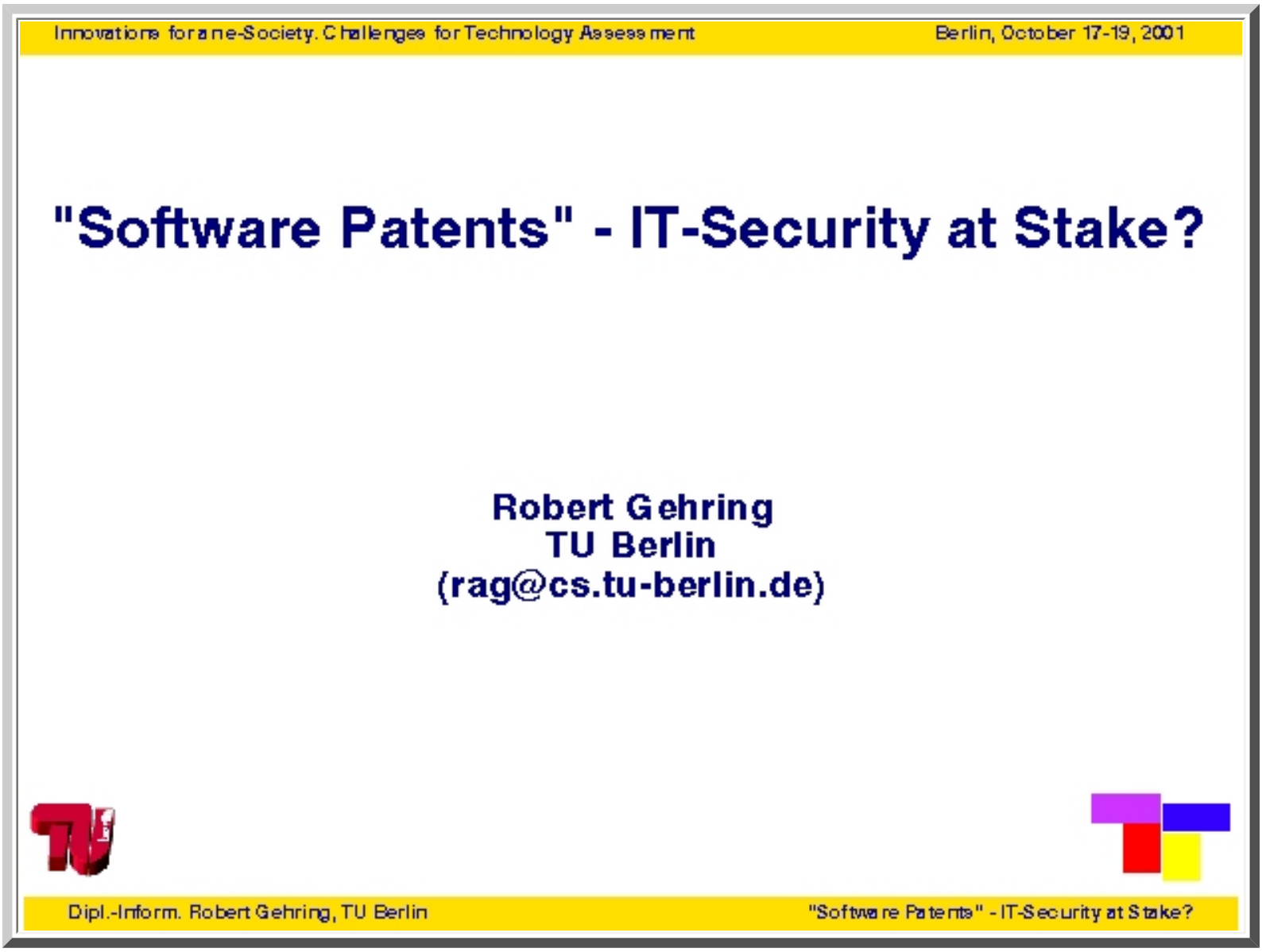
Table of Contents

Recommendation

Reference

View Text Version

"Software Patents" - IT-Security at Stake?



Notes:

Introduction

There is an ongoing legislative process of enhancing intellectual

property rights in the fields of copyright protection and patent protection.

The problem of how the security of information technology in general is affected by laws and technical measures is somewhat out of focus, at least when considering the political and legal process that lead to the new legislation.

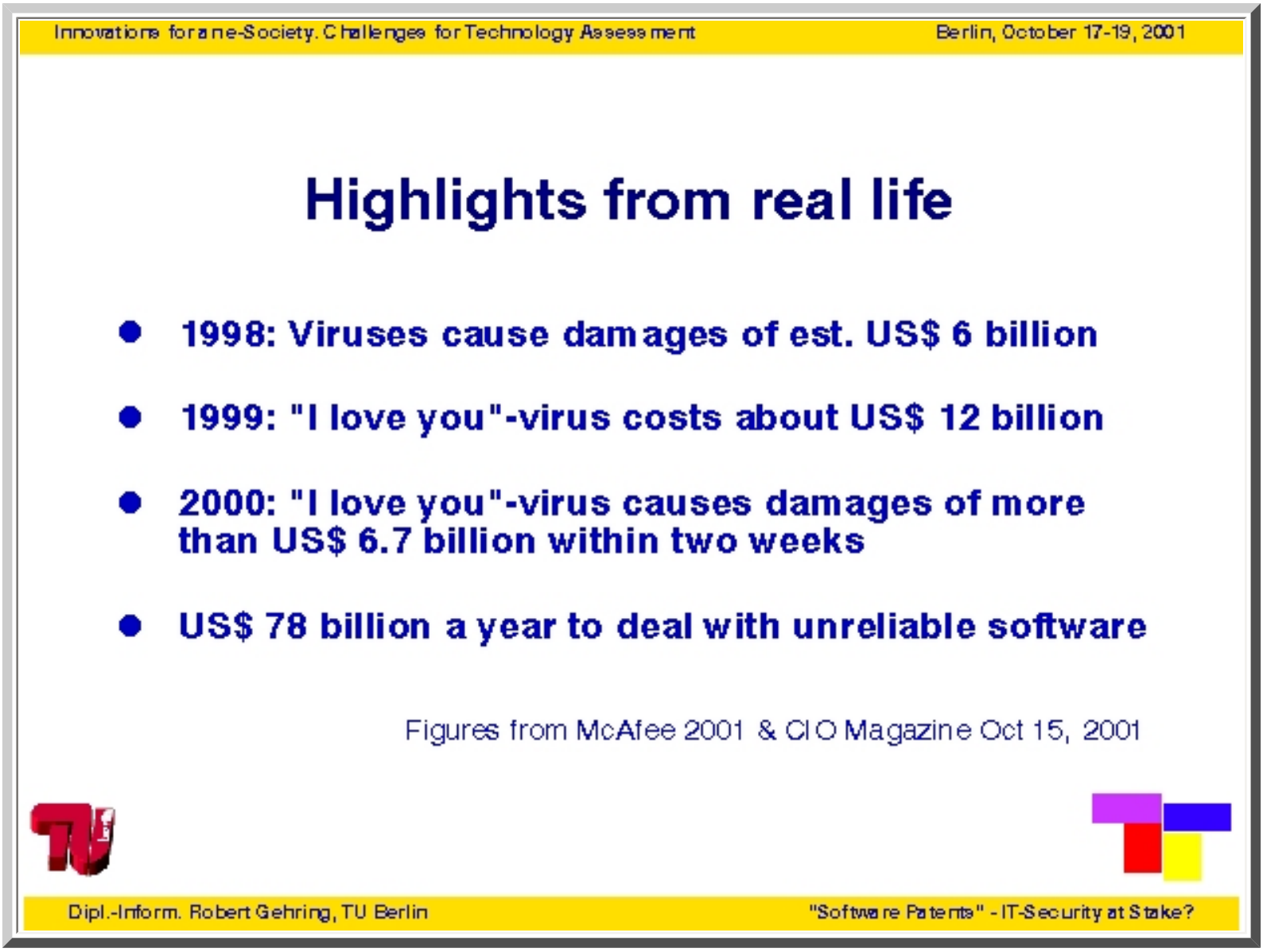
Lobbyists of the right holders put heavy pressure on the politicians to enact laws to protect their commercial interests. As the topic of security is approached, the right holders worry about the safety of their respective intellectual property assets but actually don't care much about the security of the underlying information infrastructure.

This attitude poses a critical problem for security in a networked world: Laws that protect insecure software are going to protect network insecurity at the same time.

The more the institutions of society get electronically networked the more they get at risk ¾ with unbalanced laws.

Let's take a look at the reality.

Highlights from real life



Notes:

In 1999, the so called «I love you» virus caused worldwide damages of about 12 billion dollars.

That was more than twice the damage that was counted in 1998 due to all viruses together.

In 2000, the same virus caused damages of not less than 6.7 billion dollars within 2 weeks.

And viruses are only a part of the problem.

The burden of dealing with unreliable software sums up to estimated US$78 billions a year in the U.S. industry alone.

As the following analyses will show, the insecurity of software is not due to conspiracy of fate.

"Software Patents" - IT-Security at Stake?

Robert Gehring TU Berlin (rag@cs.tu-berlin.de)

Previous slide Next slide Back to the index View Graphic Version

Notes:

Introduction

There is an ongoing legislative process of enhancing intellectual property rights in the fields of copyright protection and patent protection.

The problem of how the security of information technology in general is affected by laws and technical measures is somewhat out of focus, at least when considering the political and legal process that lead to the new legislation.

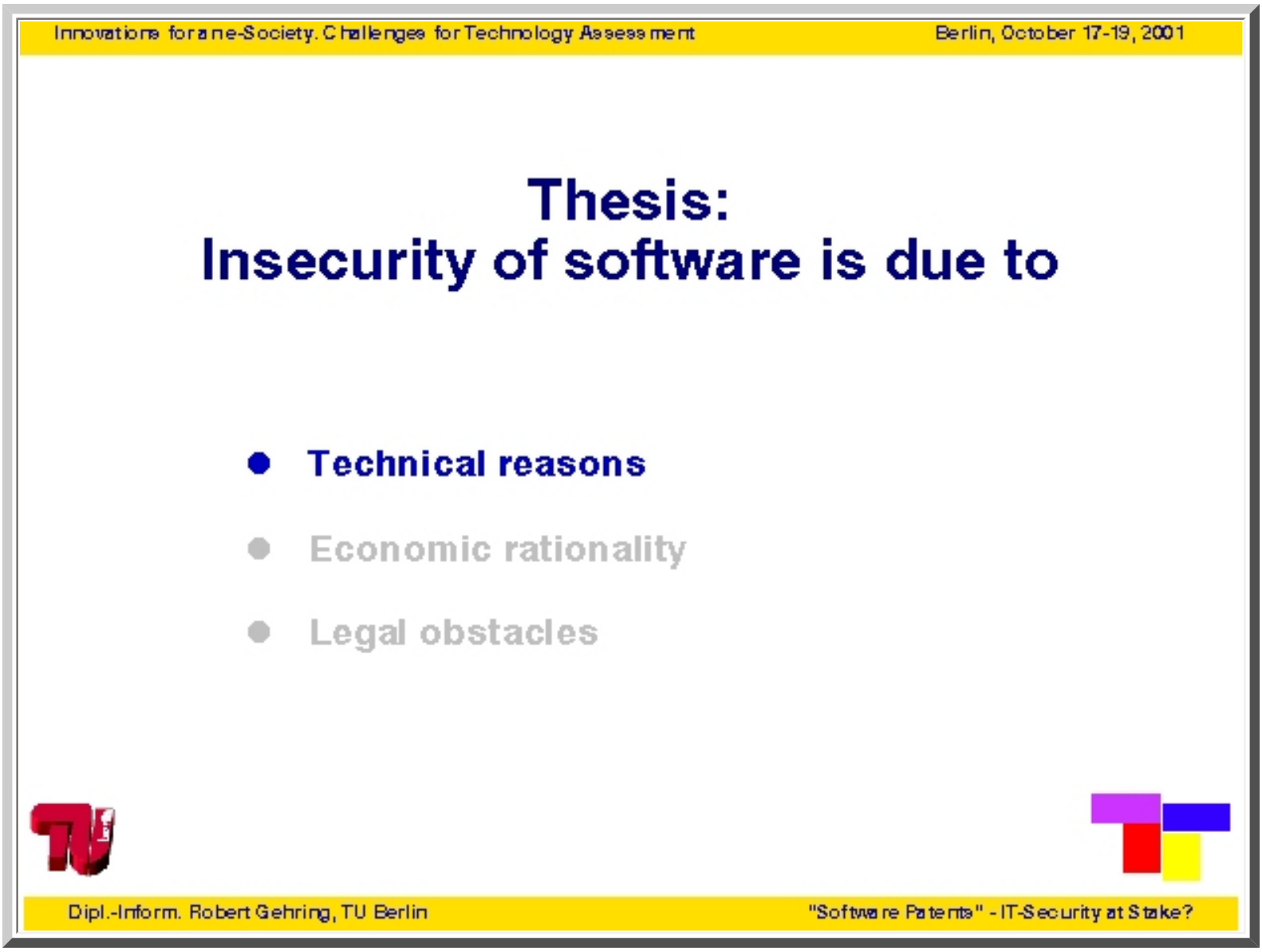
Lobbyists of the right holders put heavy pressure on the politicians to enact laws to protect their commercial interests. As the topic of security is approached, the right holders worry about the safety of their respective intellectual property assets but actually don't care much about the security of the underlying information infrastructure.

This attitude poses a critical problem for security in a networked world: Laws that protect insecure software are going to protect network insecurity at the same time.

The more the institutions of society get electronically networked the more they get at risk ¾ with unbalanced laws.

Let's take a look at the reality.

Thesis: Insecurity of software is due to



Notes:

And so goes my thesis: Insecurity of software is due to interaction of technological and legal shortcomings, fostered by economic rationality.

I will take a look at everyone of these three pillars of insecurity.

I will start with the technical reasons for insecurity.

Highlights from real life

- 1998: Viruses cause damages of est. US$ 6 billion
- 1999: "I love you"-virus costs about US$ 12 billion
- 2000: "I love you"-virus causes damages of more than US$ 6.7 billion within two weeks
- US$ 78 billion a year to deal with unreliable software

Figures from McAfee 2001 & CIO Magazine Oct 15, 2001

Previous slide Next slide Back to the index View Graphic Version

Notes:

In 1999, the so called «I love you» virus caused worldwide damages of about 12 billion dollars.

That was more than twice the damage that was counted in 1998 due to all viruses together.

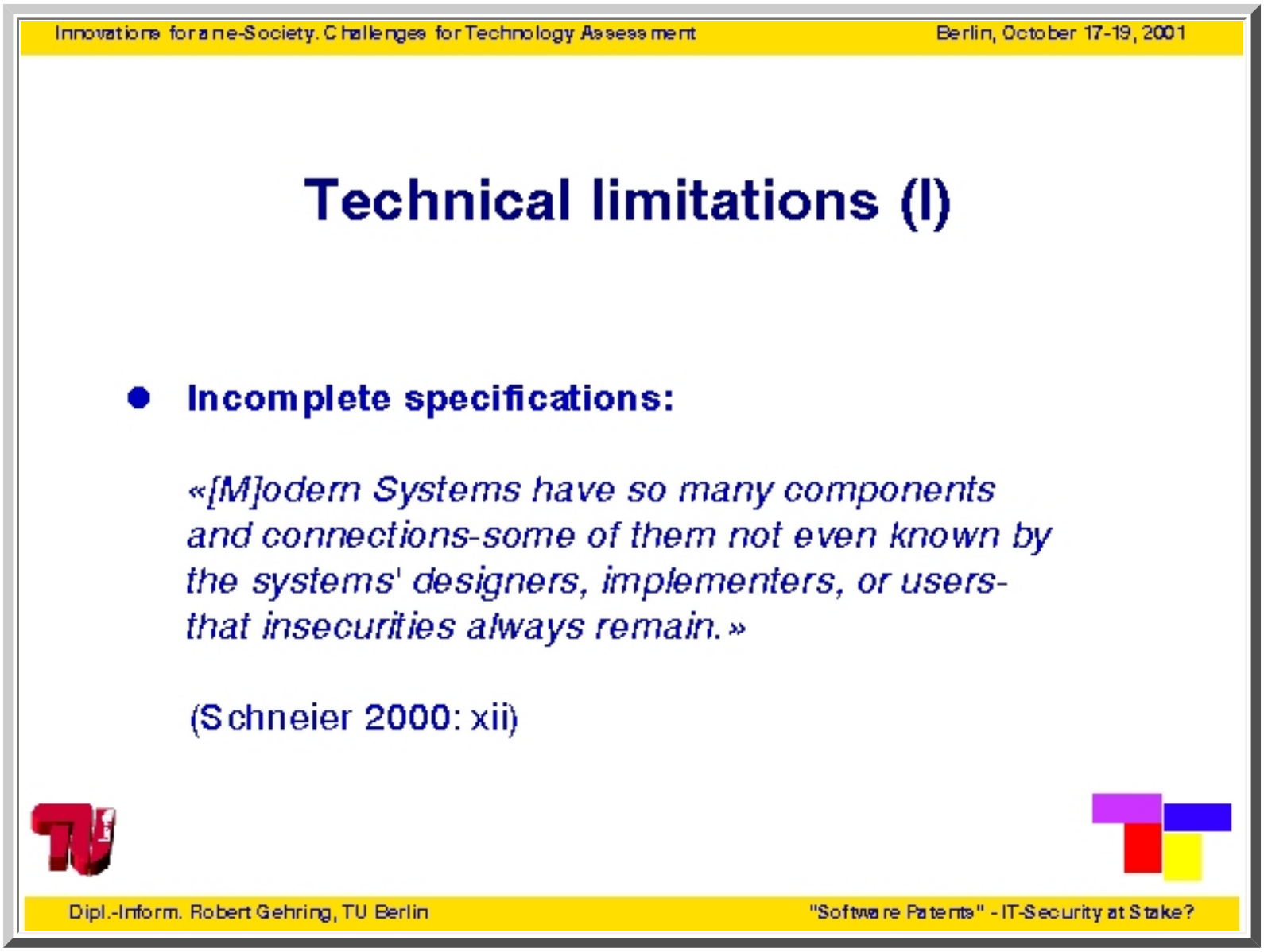
In 2000, the same virus caused damages of not less than 6.7 billion dollars within 2 weeks.

And viruses are only a part of the problem.

The burden of dealing with unreliable software sums up to estimated US$78 billions a year in the U.S. industry alone.

As the following analyses will show, the insecurity of software is not due to conspiracy of fate.

Technical limitations (I)



Notes:

The classical software development process has inherent limitations.

The first and most important limitation can bee seen in the incompleteness of functional specifications.

A functional specification serves as the blueprint for writing software.

It contains the instructions about what code the programmers will have to write.

Additionally, test instructions are derived of the functional specification.

Measured by real world conditions, a functional specification is sometimes incomplete with respect to functional requirements. And it is nearly always incomplete when it comes to security considerations.

Thesis: Insecurity of software is due to

- Technical reasons
- Economic rationality
- Legal obstacles

Previous slide Next slide Back to the index View Graphic Version

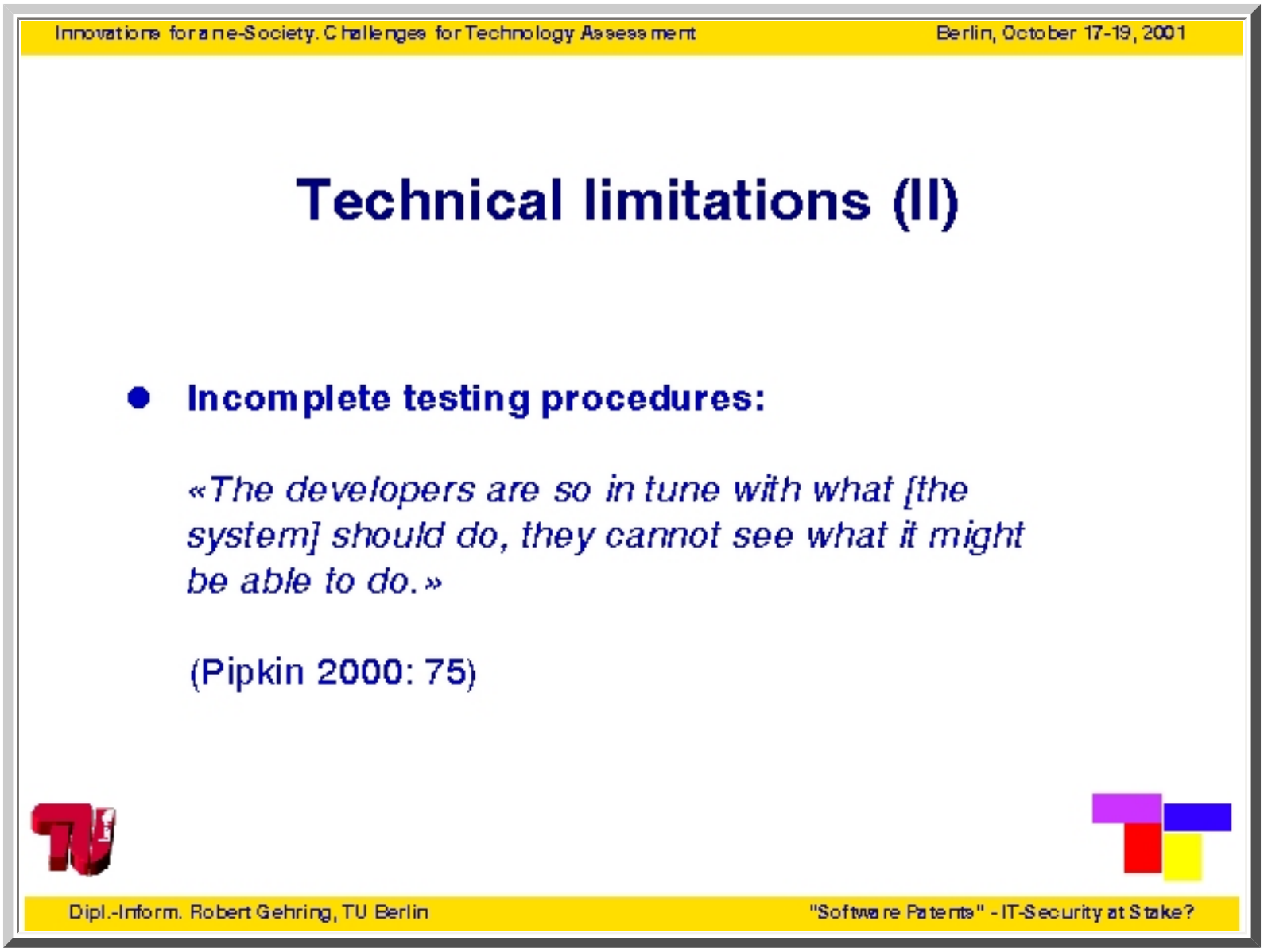
Notes:

And so goes my thesis: Insecurity of software is due to interaction of technological and legal shortcomings, fostered by economic rationality.

I will take a look at everyone of these three pillars of insecurity.

I will start with the technical reasons for insecurity.

Technical limitations (II)



Notes:

A software that is written in perfect compliance with its functional specification and has passed all the necessary tests is called correct.

But testing procedures are incomplete themselves.

No serious programmer would claim that a certain piece of software is secure because it is tested to be correct.

Technical limitations (I)

- Incomplete specifications: «[M]odern Systems have so many components and connections-some of them not even known by the systems' designers, implementers, or users-that insecurities always remain.» (Schneier 2000: xii)

Previous slide Next slide Back to the index View Graphic Version

Notes:

The classical software development process has inherent limitations.

The first and most important limitation can bee seen in the incompleteness of functional specifications.

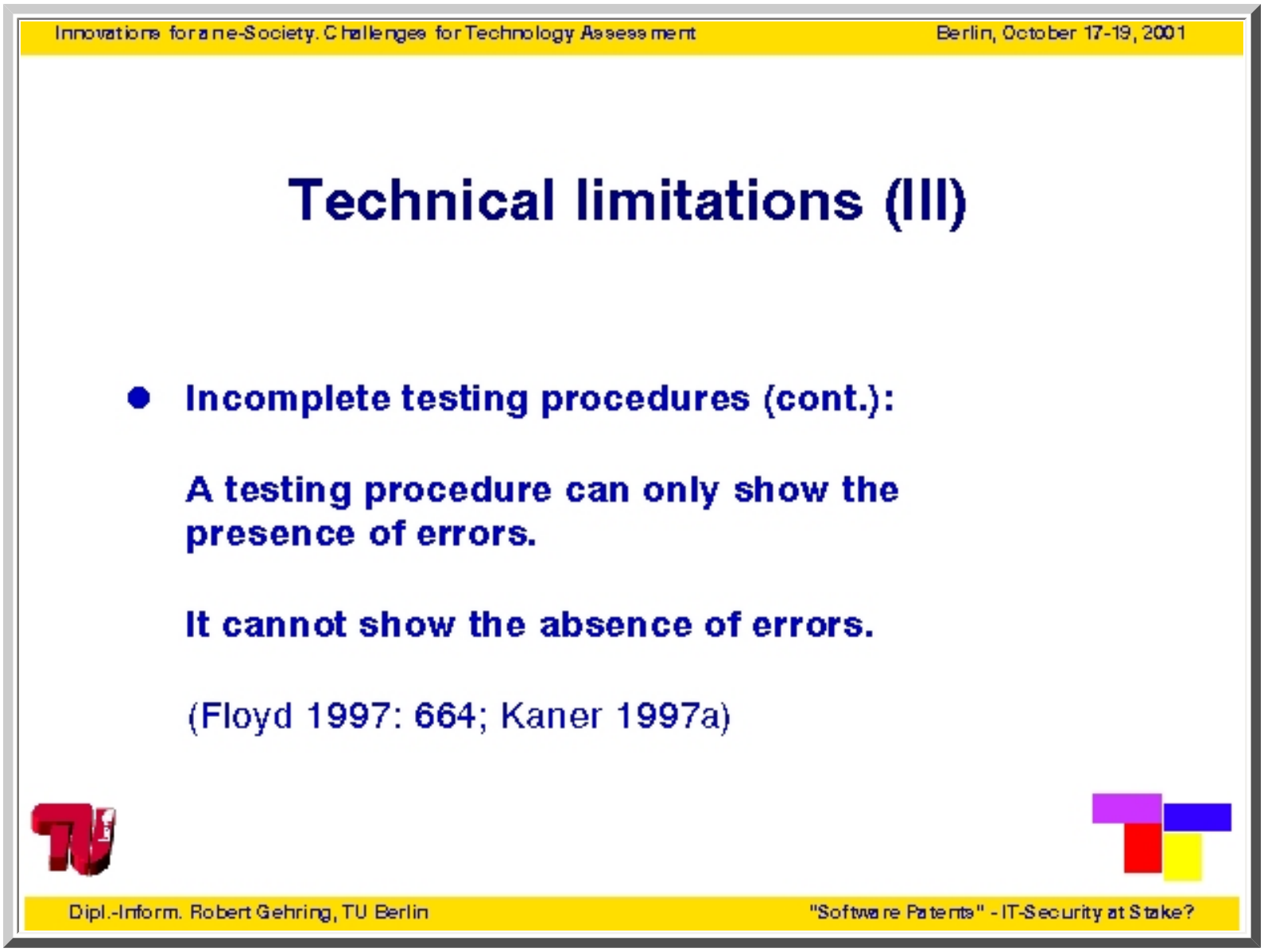
A functional specification serves as the blueprint for writing software.

It contains the instructions about what code the programmers will have to write.

Additionally, test instructions are derived of the functional specification.

Measured by real world conditions, a functional specification is sometimes incomplete with respect to functional requirements. And it is nearly always incomplete when it comes to security considerations.

Technical limitations (III)



Notes:

And testing has limitations of its own.

It is true, a lot of mistakes are detected during the testing

procedures. But not all errors can be found through testing procedures.

The testing procedure can only show the presence of errors in the program. But it cannot show the absence of errors.

Technical limitations (II)

- Incomplete testing procedures: «The developers are so in tune with what [the system] should do, they cannot see what it might be able to do.» (Pipkin 2000: 75)

Previous slide Next slide Back to the index View Graphic Version

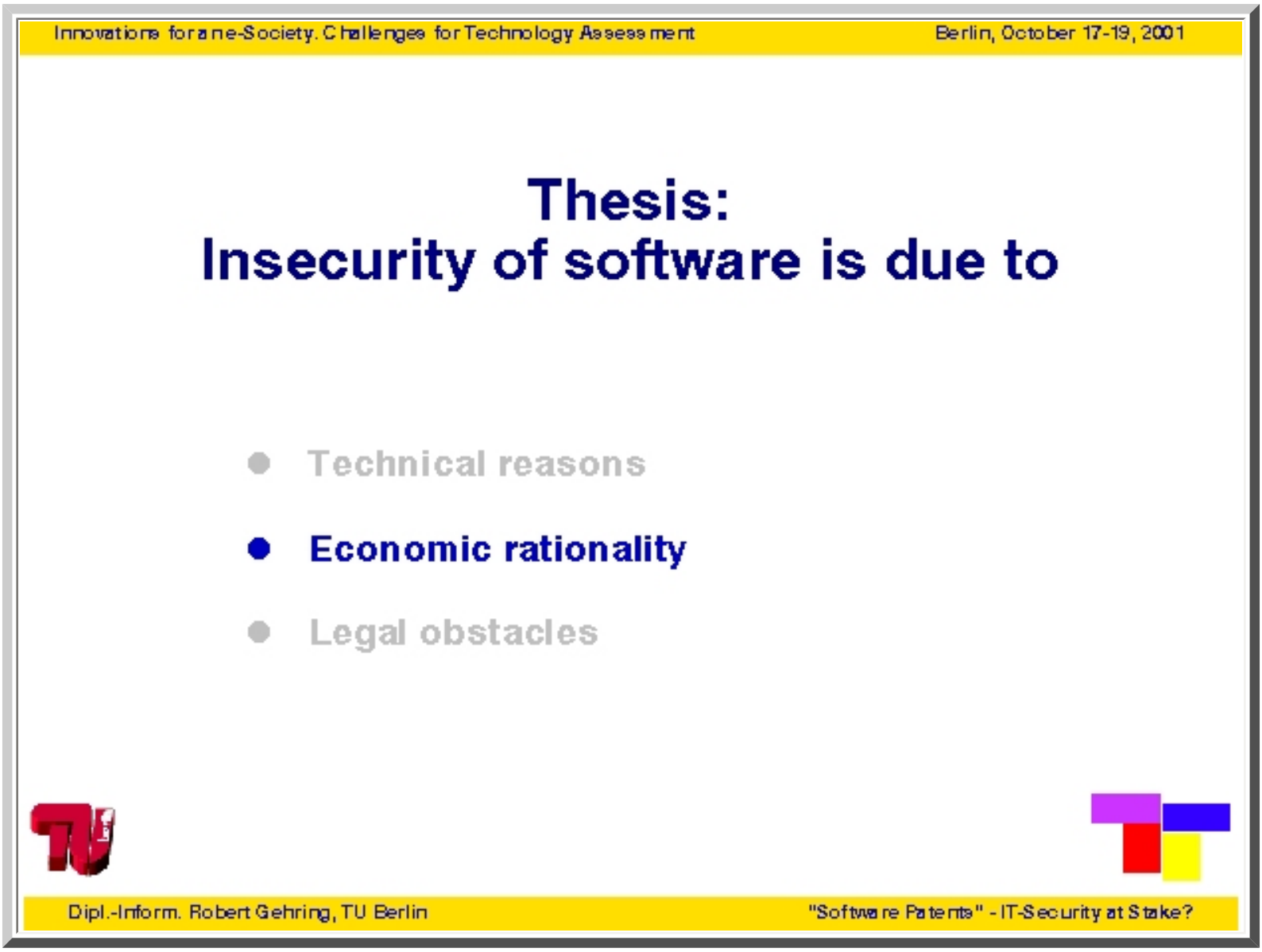
Notes:

A software that is written in perfect compliance with its functional specification and has passed all the necessary tests is called correct.

But testing procedures are incomplete themselves.

No serious programmer would claim that a certain piece of software is secure because it is tested to be correct.

Thesis: Insecurity of software is due to



Notes:

So far in short about the technical limitations.

We continue with the economic reasons for insecure software and that means to look at the economic rationality

of software producers.

Technical limitations (III)

- Incomplete testing procedures (cont.): A testing procedure can only show the presence of errors. It cannot show the absence of errors. (Floyd 1997: 664; Kaner 1997a)

Previous slide Next slide Back to the index View Graphic Version

Notes:

And testing has limitations of its own.

It is true, a lot of mistakes are detected during the testing procedures. But not all errors can be found through testing procedures.

The testing procedure can only show the presence of errors in the program. But it cannot show the absence of errors.

Economic reasons (I)

Notes:

---

(1) Proprietary software is a product for a market. A software vendor wants to make profit with it's product.

A product is only of value to its seller when a buyer can be found.
Prices are limited by market demand.

But testing and debugging software is expensive. Sophisticated testing and debugging will drive up the price for the software and cut the profits.

If the expected costs of liability for distributing unsafe software in sum are lower than the expected costs of a more complete testing and debugging process, to deliver an unsafe product will be preferred by the software producer.

That is simple economic behavior.

(2) The software market shows strong asymmetry: The vendors know about the quality of their products before selling them.

The customers learn about the quality of the software only after they have bought it and tried out.

Thus, quality considerations cannot have a great influence.

(3) And software is an experience good within a market with strong network externalities.

The positive feedback from the network externalities usually works to the advantage of the largest supplier.

Incentives to be the first on the market and to establish one's own products as de facto standards are very high.

If lowering security precautions gives an advantage, dominant market players will do so.

Thesis: Insecurity of software is due to

- Technical reasons
- Economic rationality
- Legal obstacles

Previous slide Next slide Back to the index View Graphic Version

Notes:

So far in short about the technical limitations.

We continue with the economic reasons for insecure software and that means to look at the economic rationality of software producers.

Economic reasons (II)

Notes:

Since the service capacity of a mass market software producer is limited regarding the millions of customers, the amount of available service is limited.

And scarce goods are pricey. Thus, only a minority of customers can afford to buy the necesserary service.

Looking at the other end of the software market, to the vendors of tailor-made software, the findings are by no means better.

Their business model includes profits from selling service and support for unreliable products.

Economic reasons (I)

- Proprietary software is a product for a market. Avoiding costs -for testing and debugging- is economically rational.
- The software market supplies asymmetric information to vendors and customers (a "market for lemons", Akerlof 1970).
- Network externalities (Shapiro/Varian 1999) have a great (a bad) influence.

Previous slide Next slide Back to the index View Graphic Version

Notes:

---

(1) Proprietary software is a product for a market. A software vendor wants to make profit with it's product.

A product is only of value to its seller when a buyer can be found. Prices are limited by market demand.

But testing and debugging software is expensive. Sophisticated testing and debugging will drive up the price for the software and cut the profits.

If the expected costs of liability for distributing unsafe software in sum are lower than the expected costs of a more complete testing and debugging process, to deliver an unsafe product will be preferred by the software producer.

That is simple economic behavior.

(2) The software market shows strong asymmetry: The vendors know about the quality of their products before selling them.

The customers learn about the quality of the software only after they have bought it and tried out.

Thus, quality considerations cannot have a great influence.

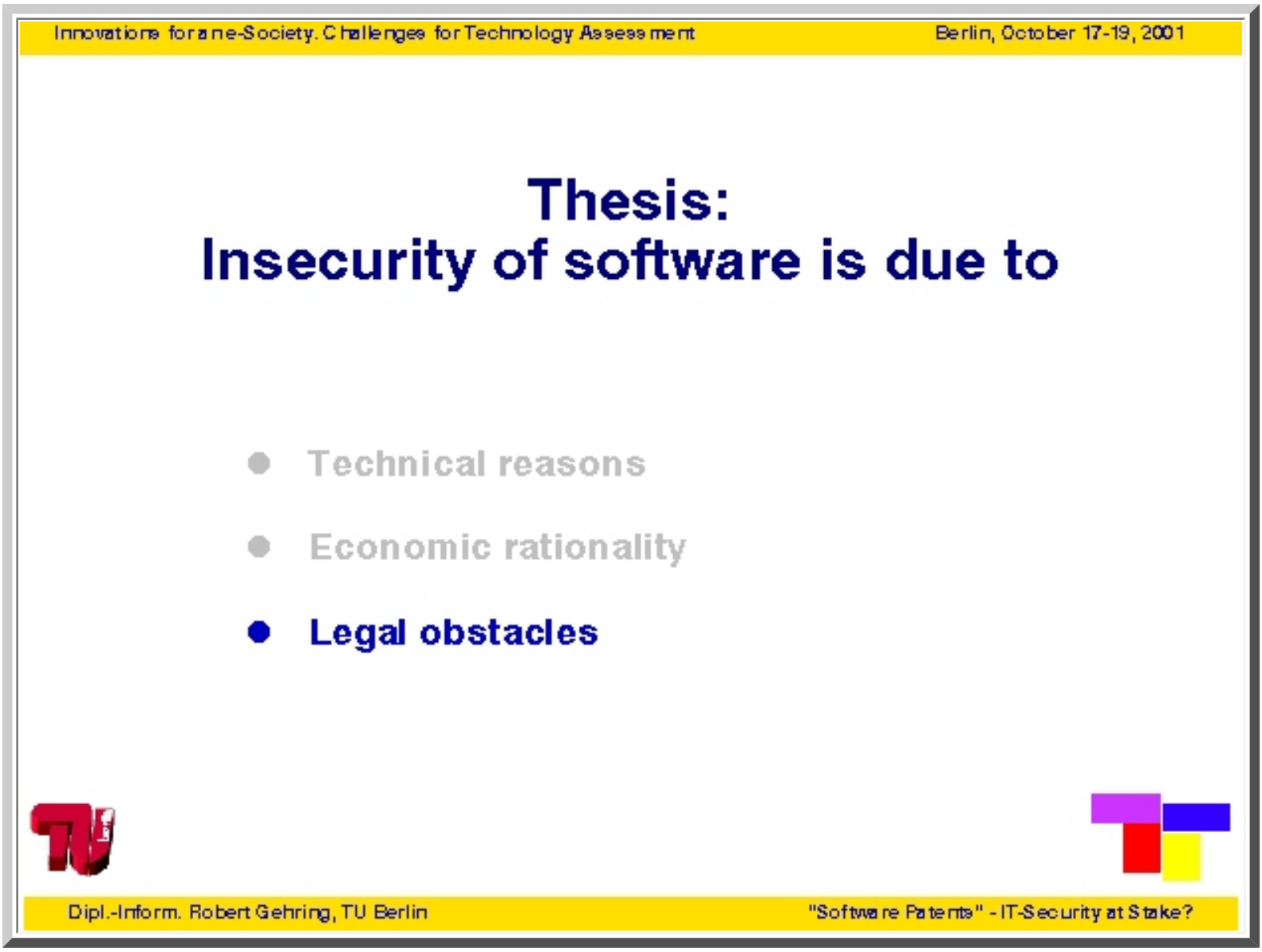
(3) And software is an experience good within a market with strong network externalities.

The positive feedback from the network externalities usually works to the advantage of the largest supplier.

Incentives to be the first on the market and to establish one's own products as de facto standards are very high.

If lowering security precautions gives an advantage, dominant market players will do so.

Thesis: Insecurity of software is due to



Notes:

Besides the technical and economic reasons for insecuriy, there are legal obstacles to better software reliability.

Economic reasons (II)

-
- Mass market software vendors have limited service capacities.
- Vendors of tailor-made software derive profits from «maintenance ... successive updates, patches and versions» (Levinson 2001).

Previous slide Next slide Back to the index View Graphic Version

Notes:

Since the service capacity of a mass market software producer is limited regarding the millions of customers, the amount of available service is limited.

And scarce goods are pricey. Thus, only a minority of customers can afford to buy the necesserary service.

Looking at the other end of the software market, to the vendors of tailor-made software, the findings are by no means better.

Their business model includes profits from selling service and support for unreliable products.

Copyright shortcomings

Notes:

(1) Copyright protection for software contains a broad ban -with only a few exceptions, e.g. for interoperability- on reverse engineering.

There is no exception for security inspection and/or enhancement.

There is no exception for to remove minor errors.

Self-help of software users regularly is declared an illegal activity.

From a legal point of view, software is treated as some kind of literary information, a kind of written speech.

And in general you cannot be held liable for distribution of literary information that contains errors. That rule applies not only to books but to software too.

There is no effective liability law to be applied in cases involving mass market software.

Thesis: Insecurity of software is due to

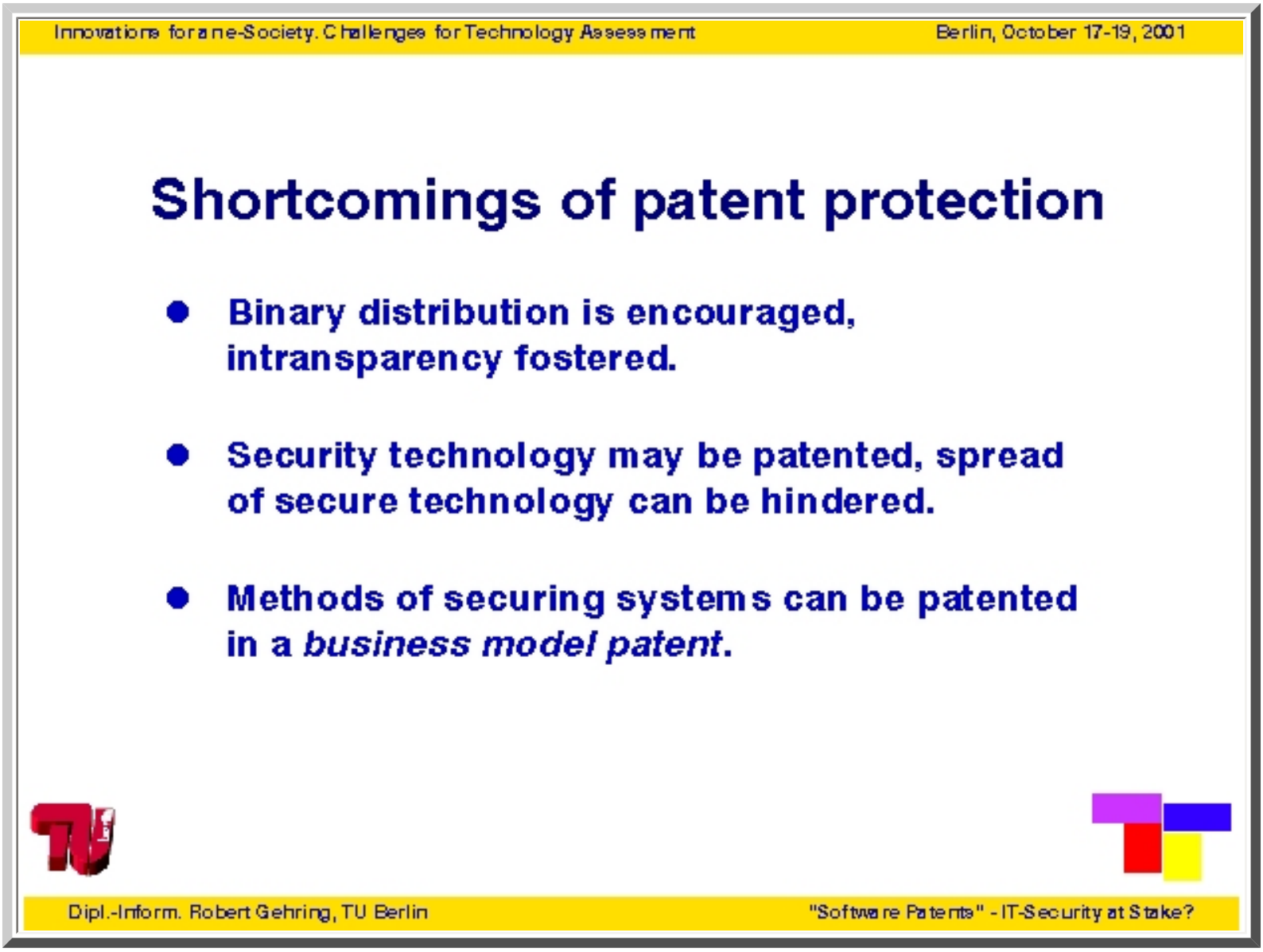
- Technical reasons
- Economic rationality
- Legal obstacles

Previous slide Next slide Back to the index View Graphic Version

Notes:

Besides the technical and economic reasons for insecuriy, there are legal obstacles to better software reliability.

Shortcomings of patent protection



Notes:

The second important legal hindrance to the enhancement of quality and security has to be seen in the increasing patent protection for software.

Patent laws give the patent holders the exclusive rights to the patented technology - in every possible implementation.

Three points show the problems connected to patent protection.

(1) Binary distribution of software will be preferred because possible patent infringment can be hidden. Thus, transparency with respect to the quality of code is decreased.

(2) Secure technology itself may be patented. Thereby the fast spread of secure technology can be delayed.

(3) Business models with a core idea of securing systems may be patented. Actually, it already happened.

One has to acquire a license in order to make systems safe or to fix security flaws in a certain way.

Copyright shortcomings

- Ban of reverse engineering.
- Repair is unlawful.
- Security enhancement is unlawful.
- No liability for written speech.

Previous slide Next slide Back to the index View Graphic Version

Notes:

---

(1) Copyright protection for software contains a broad ban -with only a few exceptions, e.g. for interoperability- on reverse engineering.

There is no exception for security inspection and/or enhancement.

There is no exception for to remove minor errors.

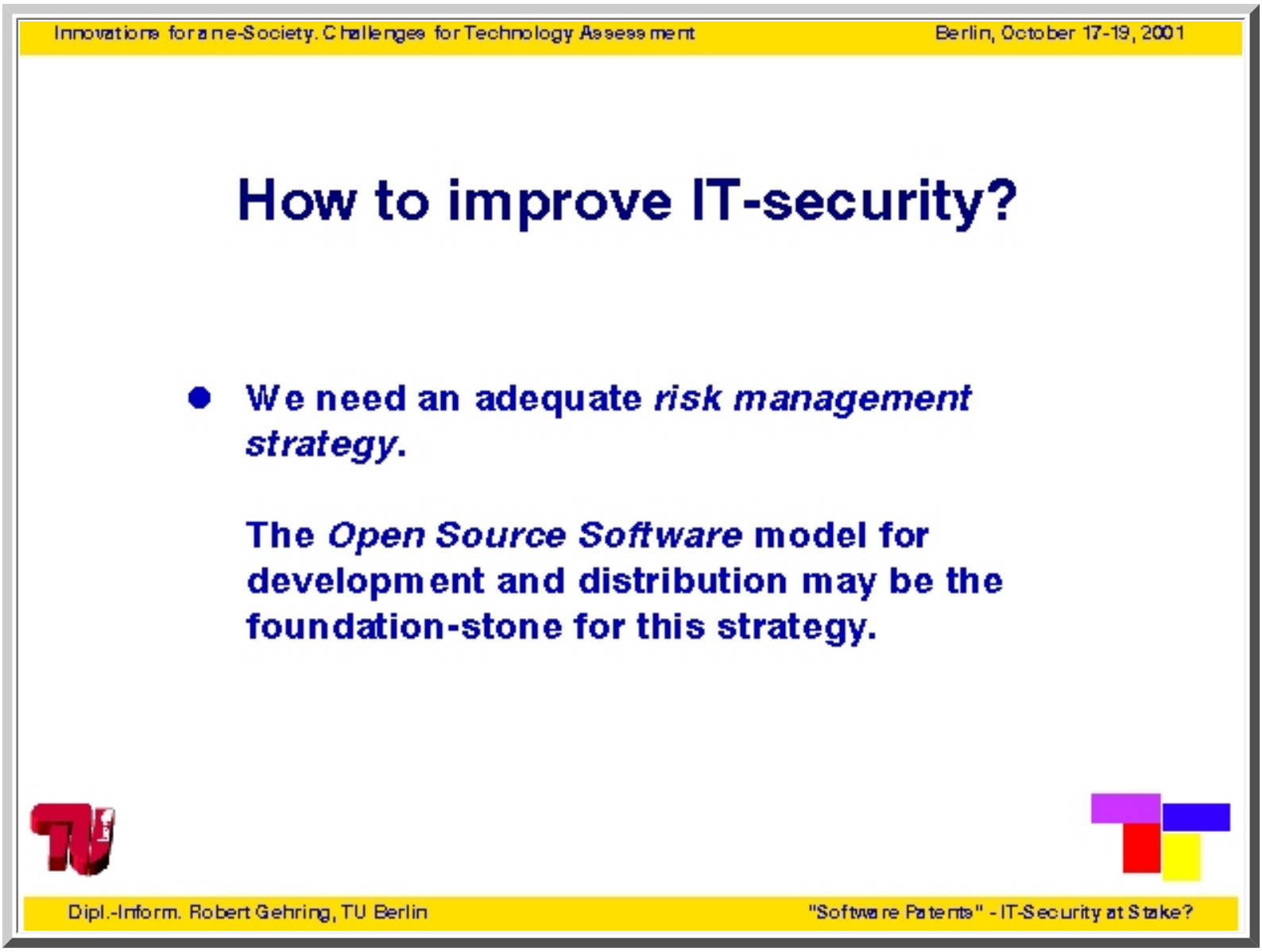
Self-help of software users regularly is declared an illegal activity.

From a legal point of view, software is treated as some kind of literary information, a kind of written speech.

And in general you cannot be held liable for distribution of literary information that contains errors. That rule applies not only to books but to software too.

There is no effective liability law to be applied in cases involving mass market software.

How to improve IT-security?



Notes:

How to improve the unsatisfying situation?

In view of the error-prone development process which leads to faulty

software, which in turn leads to insecure systems, it is time to ask for an adequate risk management strategy.

Such a risk management strategy has to be constructed in a manner that it will reduce the risks coupled with the use of software in the long term.

The best method we know so far to enhance the quality of software is the deployment of well tested standard components combined with a process of peer review by experts.

We need a better peer review process than what a single software producer is able to provide.

It must be scalable to the magnitude of the ever more ubiquitous internet. It has to have short reaction time cycles, and it must be transparent.

Here comes the open source model into play.

Shortcomings of patent protection

- Binary distribution is encouraged, intransparency fostered.
- Security technology may be patented, spread of secure technology can be hindered.
- Methods of securing systems can be patented in a business model patent.

Previous slide Next slide Back to the index View Graphic Version

Notes:

The second important legal hindrance to the enhancement of quality and security has to be seen in the increasing patent protection for software.

Patent laws give the patent holders the exclusive rights to the patented technology - in every possible implementation.

Three points show the problems connected to patent protection.

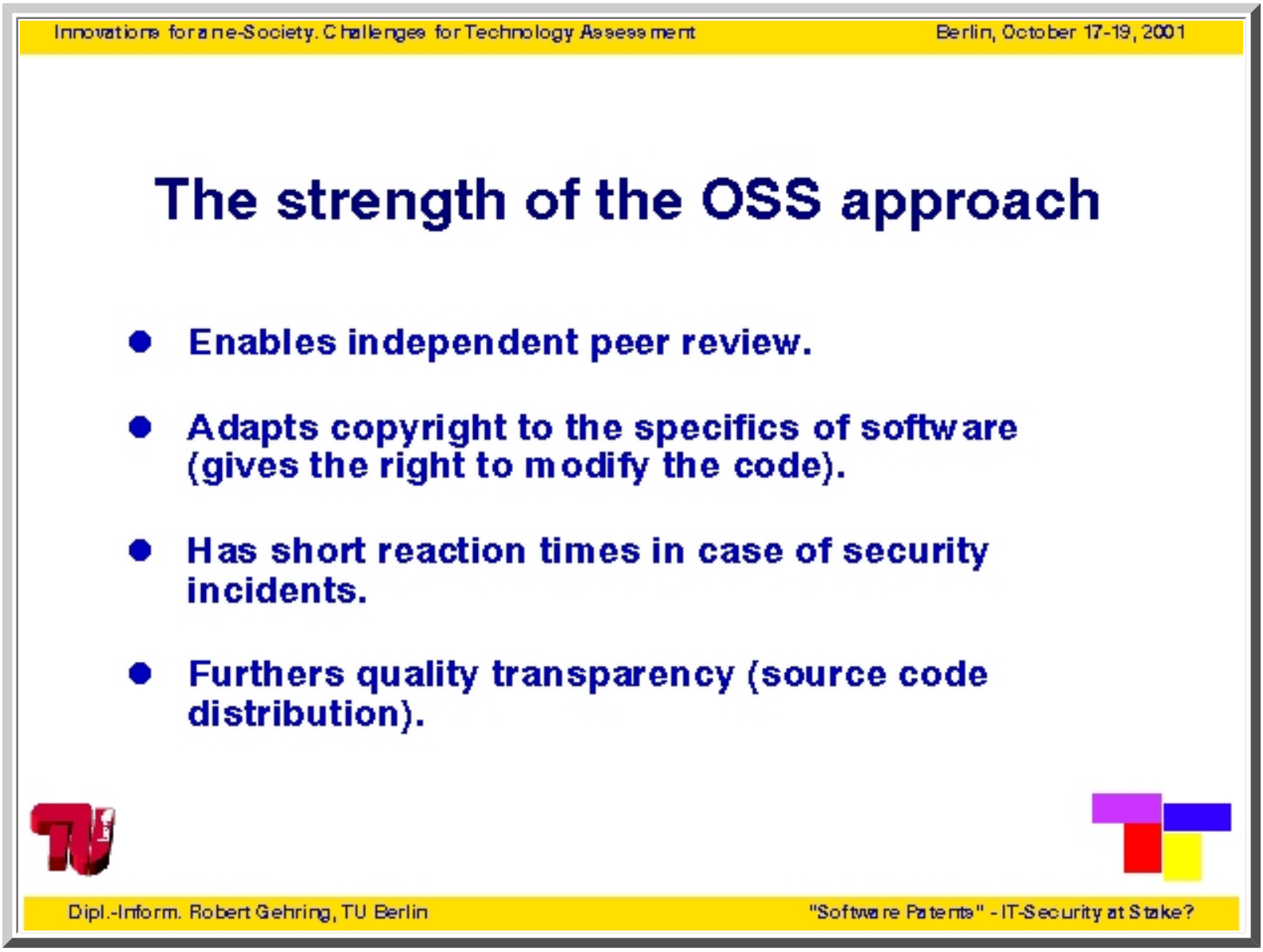
(1) Binary distribution of software will be preferred because possible patent infringment can be hidden. Thus, transparency with respect to the quality of code is decreased.

(2) Secure technology itself may be patented. Thereby the fast spread of secure technology can be delayed.

(3) Business models with a core idea of securing systems may be patented. Actually, it already happened.

One has to acquire a license in order to make systems safe or to fix security flaws in a certain way.

The strength of the OSS approach



Notes:

To the current knowledge, there is only one peer review process that fulfills the mentioned requirements and at the same time supplies the basis of a security process.

This is the open source software development model.

The open source software development process is based on the unrestricted availability of the software's source code.

Software developers all over the world take this code basis, test, enhance and secure it.

Since the source code of the program is publicly available, there is no need for reverse engineering to be able to understand how the program works.

This is not the solution but the decisive prerequisite a number of security experts demand in order to make secure systems available.

Above all, the available source code enables users at work and at home to fix security weaknesses as soon as they are detected.

There are more technical advantages of software that is available in source code.

But what about the economics of security?

How are the earlier recognized economic shortcomings of proprietary, closed source software resolved with open source software?

Firstly, the availability and use of open standards removes the bias in favor of a dominant market player and its proprietary technology.

Secondly, the market transparency grows.

Thirdly, confronted with competing offers of quality open source software, vendors of proprietary software may well be forced to undertake efforts to increase the quality of their products.

Otherwise, they may lose the competition in the long run.

But this model is in danger.

The gold-rush in software patenting we could see grow over the last years may well stall what looks yet so promising.

How to improve IT-security?

- We need an adequate risk management strategy. The Open Source Software model for development and distribution may be the foundation-stone for this strategy.

Previous slide Next slide Back to the index View Graphic Version

Notes:

How to improve the unsatisfying situation?

In view of the error-prone development process which leads to faulty software, which in turn leads to insecure systems, it is time to ask for an adequate risk management strategy.

Such a risk management strategy has to be constructed in a manner that it will reduce the risks coupled with the use of software in the long term.

The best method we know so far to enhance the quality of software is the deployment of well tested standard components combined with a process of peer review by experts.

We need a better peer review process than what a single software producer is able to provide.

It must be scalable to the magnitude of the ever more ubiquitous internet. It has to have short reaction time cycles, and it must be transparent.

Here comes the open source model into play.

The patent threat

Notes:

There are serious problems connected to the patent protection for software.

Open source software developers are particularly vulnerable to patent litigation because the code of the programs is open to everyone to inspect for patent infringement.

On the contrary, the vendors of proprietary software, which distribute their products in binary form, are protected against such search by the ban of reverse engineering in copyright laws.

And the uninhibited patenting of nearly every trivial idea if only expressed in patent lawyers terms and implemented in software has led to the much criticized situation that complex software normally should not be written any longer without extensive patent search.

This is a job for a patent specialist, not for the average programmers.

Large software producers do have a patent department at their hands. The average open source programmer doesn't have such support.

Independent developers, small and medium enterprises which develop and distribute open source software are easily being driven out of the market by larger competitors that can afford a patent litigation suit.

The strength of the OSS approach

- Enables independent peer review.
- Adapts copyright to the specifics of software (gives the right to modify the code).
- Has short reaction times in case of security incidents.
- Furthers quality transparency (source code distribution).

Previous slide Next slide Back to the index View Graphic Version

Notes:

To the current knowledge, there is only one peer review process that fulfills the mentioned requirements and at the same time supplies the basis of a security process.

This is the open source software development model.

The open source software development process is based on the unrestricted availability of the software's source code.

Software developers all over the world take this code basis, test, enhance and secure it.

Since the source code of the program is publicly available, there is no need for reverse engineering to be able to understand how the program works.

This is not the solution but the decisive prerequisite a number of security experts demand in order to make secure systems available.

Above all, the available source code enables users at work and at home to fix security weaknesses as soon as they are detected.

There are more technical advantages of software that is available in

source code.

But what about the economics of security?

How are the earlier recognized economic shortcomings of proprietary, closed source software resolved with open source software?

Firstly, the availability and use of open standards removes the bias in favor of a dominant market player and its proprietary technology.

Secondly, the market transparency grows.

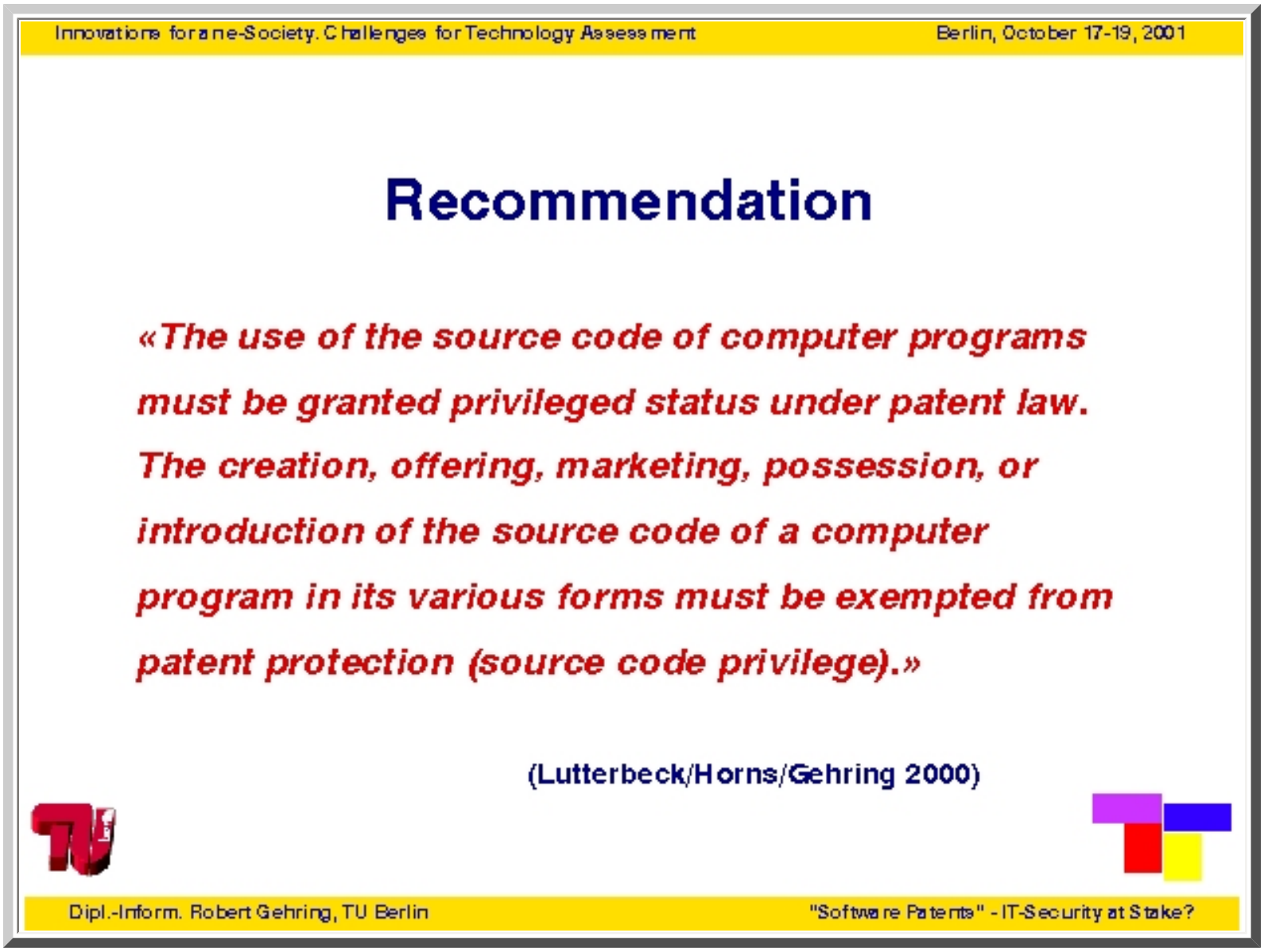
Thirdly, confronted with competing offers of quality open source software, vendors of proprietary software may well be forced to undertake efforts to increase the quality of their products.

Otherwise, they may lose the competition in the long run.

But this model is in danger.

The gold-rush in software patenting we could see grow over the last years may well stall what looks yet so promising.

Recommendation



Notes:

Perhaps, the proposal of a "source code privilege" to be included in patent laws (as presented last year in the short expertise on "Security in Information Technology and Patent Protection for Software Products:

A Contradiction?", Commissioned by the Federal Ministry of Economics and Technology), may show a way to a solution. (Lutterbeck et al. 2000)

The core proposal suggests (Recommendation PP-1):

«The use of the source codes of computer programs must be granted privileged status under patent law. The creation, offering, marketing, possession, or introduction of the source code of a computer program in its various forms must be exempted from patent protection (source code privilege).»

So I come to an end and want to close my presentation with a wish directed to all those who are interested in matters of IT security.

We should not allow the open source software development model to be destroyed in the name of the business interests of patent holders.

We should pay the required attention to the security needs of today's information infrastructures.

The patent threat

- Source code may be inspected (by competitors) for patent infringement - binary code not.
- Extensive patent search requires the help of patent lawyers. Independent developers cannot afford this.
- Independent developers, small and medium enterprises are not in the position to defend against unreasonable patent litigation.

Previous slide Next slide Back to the index View Graphic Version

Notes:

There are serious problems connected to the patent protection for software.

Open source software developers are particularly vulnerable to patent litigation because the code of the programs is open to everyone to inspect for patent infringement.

On the contrary, the vendors of proprietary software, which distribute their products in binary form, are protected against such search by the ban of reverse engineering in copyright laws.

And the uninhibited patenting of nearly every trivial idea if only expressed in patent lawyers terms and implemented in software has led to the much criticized situation that complex software normally should not be written any longer without extensive patent search.

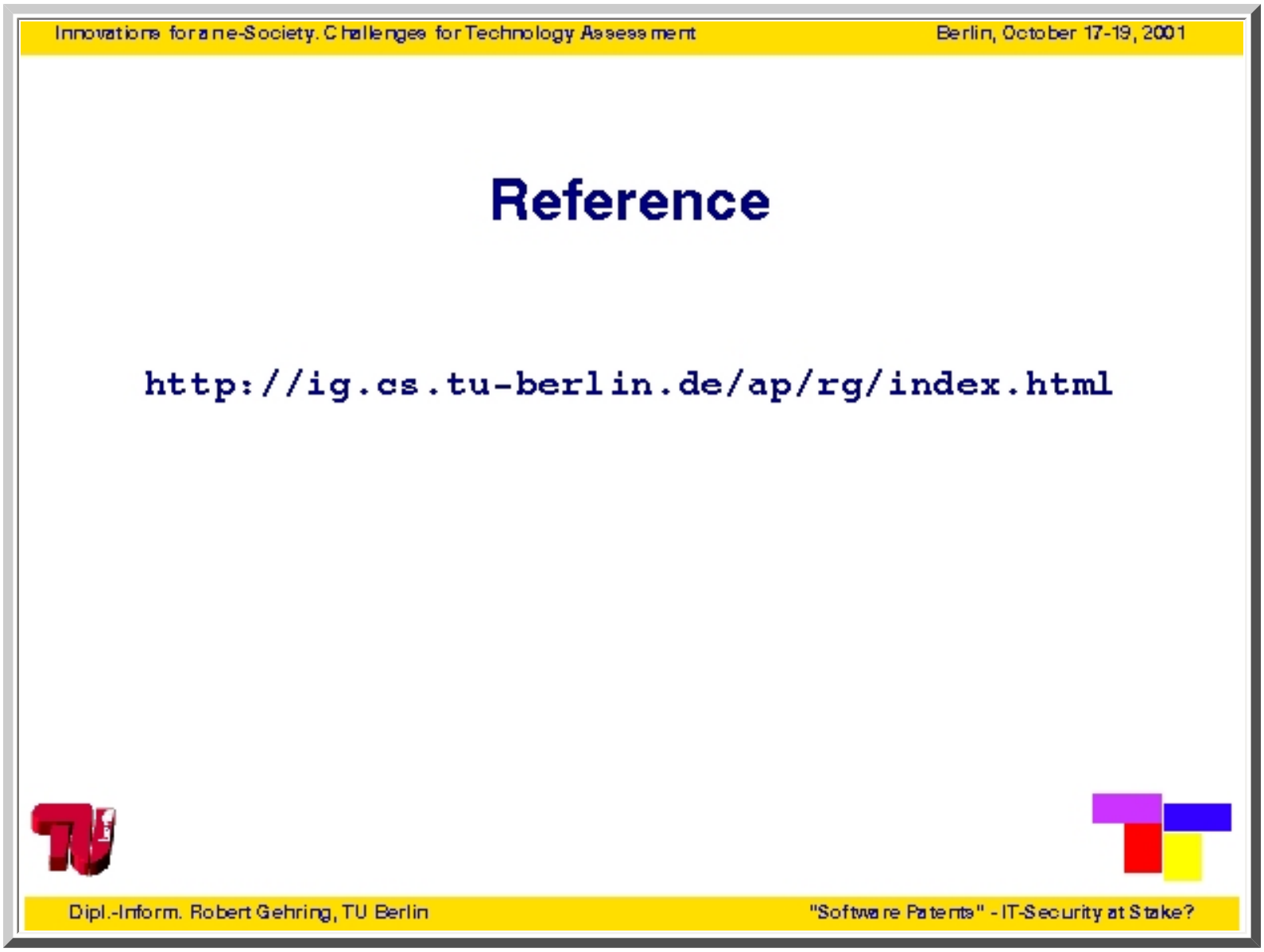
This is a job for a patent specialist, not for the average programmers.

Large software producers do have a patent department at their hands. The average open source programmer doesn't have such support.

Independent developers, small and medium enterprises which develop and

distribute open source software are easily being driven out of the market by larger competitors that can afford a patent litigation suit.

Reference



Notes:

This presentation, a short and full a length version of the conference paper can be found online, at: http://ig.cs.tu-berlin.de/ap/rg/index.html

Recommendation

«The use of the source code of computer programs must be granted privileged status under patent law. The creation, offering, marketing, possession, or introduction of the source code of a computer program in its various forms must be exempted from patent protection (source code privilege).» (Lutterbeck/Horns/Gehring 2000)

Previous slide Next slide Back to the index View Graphic Version

Notes:

Perhaps, the proposal of a "source code privilege" to be included in patent laws (as presented last year in the short expertise on "Security in Information Technology and Patent Protection for Software Products: A Contradiction?", Commissioned by the Federal Ministry of Economics and Technology), may show a way to a solution. (Lutterbeck et al. 2000)

The core proposal suggests (Recommendation PP-1):

«The use of the source codes of computer programs must be granted privileged status under patent law. The creation, offering, marketing, possession, or introduction of the source code of a computer program in its various forms must be exempted from patent protection (source code privilege).»

So I come to an end and want to close my presentation with a wish directed to all those who are interested in matters of IT security.

We should not allow the open source software development model to be destroyed in the name of the business interests of patent holders.

We should pay the required attention to the security needs of today's information infrastructures.

Reference

http://ig.cs.tu-berlin.de/ap/rg/index.html

Previous slide Next slide Back to the index View Graphic Version

Notes:

This presentation, a short and full a length version of the conference paper can be found online, at: http://ig.cs.tu-berlin.de/ap/rg/index.html

"Software Patents" - IT-Security at Stake?

Table of Contents

Recommendation

Reference

View Graphic Version