Achtung:
Deutsch/Englisch-Gemisch

# Vorlesung "Softwaretechnik"
# **Dokumentation**

Stephan Salinger
(Foliensatz: Lutz Prechelt, Bernd Brügge, Allen Dutoit)

Freie Universität Berlin, Institut für Informatik
http://www.inf.fu-berlin.de/inst/ag-se/

- Arten von Dokumentation
- Qualitätseigenschaften
  - übersichtlich, präzise, korrekt
  - hilfreich
- Positive und negative Beispiele
- Prinzipien
  - Selbstdokumentation
  - Minimaldokumentation
- Rationale Management (Begründungsmanagement):
  - Issues
  - Proposals
  - Criteria
  - Arguments
  - Resolutions
- Englisch

# Wo sind wir?: Taxonomie "Die Welt der Softwaretechnik"

Freie Universität Berlin

Welt der Problemstellungen:

- Produkt (Komplexitätsprob.)
  - **Anforderungen (Problemraum)**
  - **Entwurf (Lösungsraum)**
- Prozess (psycho-soziale P.)
  - **Kognitive Beschränkungen**
  - **Mängel der Urteilskraft**
  - **Kommunikation, Koordination**
  - **Gruppendynamik**
  - **Verborgene Ziele**
  - **Fehler**

Welt der Lösungsansätze:

- Technische Ansätze ("hart")
  - **Abstraktion**
  - **Wiederverwendung**
  - **Automatisierung**
- Methodische Ansätze ("weich")
  - **Anforderungsermittlung**
  - **Entwurf**
  - **Qualitätssicherung**
  - **Projektmanagement**

# Wo sind wir?: Dokumentation

Freie Universität Berlin

- Einsicht: Der Nutzen von SW entfaltet sich nur, wenn viele Information schriftlich weitergegeben wird
  - schon innerhalb eines Projekts, erst recht darüber hinaus

- Prinzipien:
  - **Mehrwert**: Dokumentation sollte vor allem die Information enthalten, die ohne sie nicht offensichtlich ist
  - **Nützlichkeit**: Dokumentation sollte hauptsächlich solche Information enthalten, die für Leser auch tatsächlich hilfreich ist.
  - **Lokalität**: Dokumentation sollte möglichst eng mit dem Dokumentierten verbunden sein
  - **Integration**: Dokumentation sollte zeitnah zur Entstehung des Dokumentierten angefertigt werden
  - **Redundanzarmut**: Eine Information sollte möglichst nicht an mehreren Stellen dokumentiert sein
  - **Nachvollziehbarkeit**: Interne Dokumentation soll vor allem erlauben, die Überlegungen der Erbauer/innen nachzuvollziehen

# Was ist Dokumentation?

- Definition "Dokumentation":
  All diejenige Information in Dokumenten, die nicht vorrangig dazu dient, einen Computer zu instruieren

- Beachte:
  - Dadurch gehören Kommentare im Quellcode automatisch zur Dokumentation
  - aber ebenso auch schon Bezeichner
  - UML-Diagramme sind zum Teil reine Dokumentation, zum Teil ein Gemisch wie Quellcode – je nach Verwendung

  (Es gibt auch andere Definitionen für Dokumentation)

# Arten von Dokumentation

Freie Universität Berlin

**Externe Dokumentation** (für Anwender):

- Benutzeranleitungen
  - Lernanleitungen, Bedienhandbücher, Referenzhandbücher
- Installationsanleitungen
- Administrationsanleitungen
- Versionsbeschreibung ("release notes")
  - Änderungen zur Vorversion, bekannte Mängel, etc.

**Interne Dokumentation** (für Entwickler):

- Produktdokumentation:
  - Anforderungsbeschreibungen
  - Entwurfsbeschreibungen
  - Details zur Implementierung
  - Testfallbeschreibungen
  - Beziehungen zwischen diesen
    - Nachverfolgung ("tracing") von Anford./Entwurfsentsch. bis in Implementierung und Test
  - etc.
- Prozessdokumentation:
  - Projektplan u. Teilpläne
  - Test- und Durchsichtenhistorie
  - Defekt- und Änderungshistorie
  - etc.

Andere Beteiligte: Kunden, Zertifizierungsstellen, …

# Aufgaben von Dokumentation



- Alle Dokumentation dient der Kommunikation zwischen Menschen
  1. *Präzise* Weitergabe von Information zwischen gleichzeitig Prozessbeteiligten
     - siehe Projektmanagement
  2. Weitergabe wichtiger Information von jetzt Beteiligten an *später* Beteiligte
     - interne Dokumentation für Pflege und Wartung der SW
  3. Weitergabe benötigter Information von Beteiligten an *Betroffene*
     - "externe Dokumentation"
- Aspekte von Dokumentation:
  - Aufgabe/Zweck, Zielgruppe
  - Inhalt und Stil
  - Autoren
  - Technische Repräsentation
  - Organisation der Herstellung und Pflege

# Qualitätsmerkmale von Dokumentation



- Übersichtlich
  - d.h. benötigte Information ist schnell zu finden
- Präzise
  - d.h. Angaben sind (genügend) genau, nicht (zu) vage
- Korrekt
  - d.h. Angaben stimmen mit der (SW-)Realität überein und sind nicht irreführend
- **Hilfreich**
  - **d.h. beantwortet die wirklich *wichtigen* Fragen, nicht nur irgendwelche beliebigen**
- Lässt sich mit geringem Aufwand erstellen
  - d.h. ihre Struktur ist so gewählt, dass sie möglichst einfach aus dem SW-Prozess hervorgehen kann
    - Dies steht gelegentlich im Widerspruch zu den anderen Anforderungen
    - insbesondere bei externer Dokumentation

# Wer benötigt Dokumentation? Wer fertigt Dokumentation?

Freie Universität Berlin

Wer konsumiert/produziert welche Arten von Dokumentation?

**Zielgruppen:**

- Endanwender, Systemadministratoren ("externe Dok.")
- Entwerfer (Anforderungsdok.)
- Entwickler (Anforderungsdok., Entwurfsdok.)
- Inspekteure (alle Arten von Dokument)
- Tester (Anforderungsdok., Entwurfsdok.)
- Fehlerkorrekteure, Weiterentwickler (alle Arten v. Dokument)

**Autorengruppen:**

- Analysten (Anforderungsdok.)
- Entwerfer (Anforderungsdok., Entwurfsdok.)
- Entwickler (Entwurfsdok., Implementierungsdok., ext. Dok.)
- Technische Redakteure (externe Dok.)

*[Aufzählungen sind unvollständig]*

# Probleme mit Dokumentation im SW-Prozess

Freie Universität Berlin

- Während der Erstellung einer Software werden viele Dokumentationsinhalte nur mündlich weitergegeben. Vorteile:
  - Effizient (wenn Empfängerzahl klein)
  - Erlaubt Rückfragen
  - Vermeidet bei häufigen Änderungen das Veralten der Information
- Da viele SW-Ingenieure weder gern noch gut schreiben, wird schriftliche interne Dokumentation oft vernachlässigt
- → Folge: Sie ist oft karg, oft veraltet und wenig hilfreich
- → → Folge: Sie wird wenig benutzt ("lieber nachfragen")
- → → → Folge: Sie wird noch mehr vernachlässigt

- Das funktioniert im laufenden Projekt oft ausreichend bis gut
- Nach Projektende wird es aber zu einem Riesenproblem
  - Wartungsingenieure sind ohne gute Dokumentation arm dran
  - (Und Wartung ist insgesamt oft teurer als Entwicklung!)

# Auswege aus dem Dilemma

Es gibt vier Wege, wie man mit dieser Situation umgehen kann:

1. Sie **akzeptieren** und damit leben
   - Das ist der häufigste Fall in der Praxis
2. Vollständige und aktuelle Dokumentation organisatorisch **erzwingen**
   - Das wird bei sehr sicherheitskritischen Projekten gemacht
   - Sehr teuer. Kosteneffizienz ist unklar.
3. Hohe **Selbstdokumentation** erreichen
   - ➔ dazu gleich mehr
4. Geschickte **Minimaldokumentation** bereit stellen
   - ➔ dazu gleich mehr

# Prinzip: Selbstdokumentation

Freie Universität Berlin

Prinzip:

- Entwerfe die Software mit so klaren Strukturen, dass sie auch ohne explizite zusätzliche Dokumentation verständlich ist

Vorteile:

- Spart Aufwand
- Dokumentation kann nicht veralten

Probleme:

- Sehr schwer hinzubekommen
  - Es klappt immer nur zu einem Bruchteil
- Kann nicht alle Dokumentation ersetzen
  - z.B. nicht **Anforderungen und Entwurfsbegründungen**

Fazit:

- Gute und richtige Idee, aber nur eine Teillösung

# Prinzip: Minimaldokumentation

Freie Universität Berlin

Prinzip:

- Schreibe nur denjenigen winzig kleinen Teil einer vollen Dokumentation, der am hilfreichsten ist
  - 1–5% des Ganzen bringen meist schon 50% des Nutzens

Vorteile:

- **1.** Spart Aufwand, **2.** Änderungen viel seltener nötig, **3.** Änderungen sind realistisch durchzuhalten
  - Auch als Einleitung einer ausführlichen Dokumentation geeignet

Probleme:

- Es braucht Übung, den richtigen Teil zu entdecken und gut zu beschreiben

Fazit:

- Gute und richtige Idee. Lernen!
- Konsequent anwenden!!

# Positivbeispiel Minimaldokumentation



- Paketbeschreibung **java.awt**
  - "*Contains all of the classes for creating user interfaces and for painting graphics and images. A user interface object such as a button or a scrollbar is called, in AWT terminology, a component. The Component class is the root of all AWT components. See Component for a detailed description of properties that all AWT components share.*
  - *Some components fire events when a user interacts with the components. The AWTEvent class and its subclasses are used to represent the events that AWT components can fire. See AWTEvent for a description of the AWT event model.*
  - *A container is a component that can contain components and other containers. A container can also have a layout manager that controls the visual placement of components in the container. The AWT package contains several layout manager classes and an interface for building your own layout manager. See Container and LayoutManager for more information.*"

(AWT in Java 1.2: 64+14 Klassen+Interfaces; in Java 6: 98+16; identischer Text!)

# Was ist an diesem Beispiel gut?



- "*Contains all of the classes for …*"
  - Charakterisiert das Ganze
- "*A UI object such as a button … is called … a component.*"
  - Führt das zentrale Konzept ein
- "*Some components fire events when a user interacts with the components.*"
  - Führt ein zweites Hauptkonzept ein
  - und setzt es in Beziehung zum ersten
- "*A container is a component that can contain components*"
  - Drittes Hauptkonzept und Beziehung zum ersten
- "*A container can also have a layout manager that controls…*"
  - Viertes Hauptkonzept und Beziehung zum dritten

Vorbildlich!

# Negativbeispiel Minimaldokumentation



- Paketbeschrbg **java.awt.dnd** (Java 7: 17 Klassen, 5 Interfaces)
  - "*Drag and Drop is a direct manipulation gesture found in many GUI systems that provides a mechanism to transfer information between two entities logically associated with presentation elements in the GUI.*"
  - Richtig, aber wenig hilfreich: Viel zu abstrakt!
- Klasse **java.awt.dnd.DropTarget**
  1. "*The DropTarget is associated with a Component when that Component wishes to accept drops during Drag and Drop operations.*
  2. *Each DropTarget is associated with a FlavorMap.*
  3. *The default FlavorMap hereafter designates the FlavorMap returned by SystemFlavorMap.getDefaultFlavorMap().*"
  - 1: zu abstrakt (Beispiel nötig),
    2: vage (fast ominös: Was zum Kuckuck soll eine FlavorMap?),
    3: zu detailliert (nach Satz 2 fühlt man sich fast veralbert)

# Wodurch wird Dokumentation *hilfreich*?

Freie Universität Berlin

Zwei Grundprinzipien:

- Konzepte erläutern:
  - In jeder Dokumentation gibt es eine handvoll Grundkonzepte, die stark im Mittelpunkt stehen
  - Diese sollte man explizit einführen: Was ist das? **Wozu** dient es?
  - Solche Konzepte können Daten ebenso wie Operationen betreffen
  - Wichtigste Frage für Einsteiger ist meist die nach Begründung oder Zweck
- Problemlösungen beschreiben:
  - Handeln Sie nicht einfach die Einzelteile der Reihe nach ab, sondern erklären Sie, wie man mit Ihnen die gängigsten Probleme löst
  - Beispiele, Beispiele, Beispiele!

- Beide Prinzipien gelten gleichermaßen für interne und externe Dokumentation

# Beispiel: Nicht hilfreiche externe Dokumentation

Freie Universität Berlin

Microsoft Outlook 2003 (Email- und Groupware-Klient):

- Angenommen, Sie arbeiten seit längerem damit
  - und gehen routinemäßig mit Ordnern, Nachrichten, Kontakten, Terminen und Anhängen um
- Nun stolpern Sie in der Dokumentation erstmalig über den Begriff *"persönliche Ordner-Datei"* und möchten verstehen, was das sein könnte
- Sie rufen die Hilfe auf:
  - Hilfe → Suchen → *"Persönliche Ordner-Datei"* → [Return]
- Sie erhalten 20 Antworten
  - davon spricht offenbar nur eine direkt zum Thema:

# Beispiel: Nicht hilfreiche externe Dokumentation (2)



- "***Erstellen einer Persönlichen Ordner-Datei mit Unicodeunterstützung***
  - *Zeigen Sie im Menü **Datei** auf **Neu**, und klicken Sie dann auf **Outlook-Datendatei**.*
  - *Um einen Persönlichen Ordner für Microsoft Outlook zu erstellen, der über erweiterte Speicherkapazitäten für Elemente und Ordner sowie über Unterstützung mehrsprachiger Unicodedaten verfügt, klicken Sie auf **OK**.*
  - *Geben Sie im Feld **Dateiname** einen Namen für die Datei ein, und klicken Sie auf **OK**.*
  - *Geben Sie im Feld **Name** einen Anzeigenamen für den PST-Ordner ein.*
  - *Wählen Sie alle anderen gewünschten Optionen aus, und klicken Sie auf **OK**. Der Name des zu der Datendatei gehörenden Ordners wird in der Ordnerliste angezeigt. Zum Anzeigen der Ordnerliste klicken Sie im Menü **Wechseln zu** auf **Ordnerliste**. Standardmäßig heißt der Ordner **Persönliche Ordner**."*

# Was ist an diesem Beispiel schlecht?

Freie Universität Berlin

1. Es erklärt mir wie ich etwas tue, von dem ich erst mal gern verstehen würde, ob ich es überhaupt tun will:
   - "*Erstellen einer Persönlichen Ordner-Datei mit Unicodeunterstützung*"
2. Es erläutert mir lauter Details, die ich schon weiß:
   - "*Geben Sie im Feld **Dateiname** einen Namen für die Datei ein, und klicken Sie auf **OK**.*"
   - "*Wählen Sie alle anderen gewünschten Optionen aus, und klicken Sie auf **OK**.*"
3. Es verwendet weitere Begriffe, die ich nicht kenne:
   - "*Geben Sie im Feld **Name** einen Anzeigenamen für den **PST-Ordner** ein.*"
4. Es erklärt mir aber nicht das, was ich vor allem wissen möchte: Was ich mir unter einer solchen Datei vorstellen soll
   - und zu welchem Zweck es sie gibt

# Gute externe Dokumentation kann ein wichtiger Erfolgsfaktor sein

Freie Universität Berlin

- Bei komplexer SW ist die tollste Funktionalität und Qualität nur so viel wert, wie die Dokumentation auch vermitteln kann
- Deshalb ist eine SW erst mit guter Dokumentation gut

- Beispiel: Microsoft Visual Basic (VB) gegenüber Borland Delphi
  - Die Verbreitung von Visual Basic ist um ein Vielfaches höher
  - obwohl jeder, der beides kennt, Visual Basic (zumindest als Sprache) furchtbar findet
- Der Erfolg von VB gründet sich zum großen Teil auf eine sehr gute Dokumentation:
  - "*Microsoft tools are wonderfully documented. Borland's documentation is okay but not so great. Borland seems to have the resources and cleverness to create some great tools, but not enough resources to provide the support that Microsoft does.*"
    - http://discuss.fogcreek.com/joelonsoftware/default.asp?cmd=show&ixPost=42949

# Wer schreibt externe Dokumentation?

Freie Universität Berlin

- Es gibt Menschen, die eine spezielle Ausbildung dafür haben
  1. Dokumentation so zu schreiben, dass sie die gewünschten Eigenschaften mitbringt
     - (präzise, übersichtlich, korrekt, hilfreich) und
  2. dazu eng mit Wissenslieferanten (hier: SW-Entwickler/inne/n) zu kommunizieren
- Dieser Berufszweig heißt Technische/r Redakteur/in

- Leider hapert es in der Praxis oft:
  - Keine Technischen Redakteure vorhanden
  - oder: Entwickler sind zu sehr unter Druck beim SW-Schreiben, um den Redakteuren genügend Information zu übermitteln

# Beispiel: Nicht hilfreiche interne Dokumentation

Freie Universität Berlin

- **Java 5: "*java.util Interface EventListener***
- ***All Known Subinterfaces:***
  - *Action, ActionListener, AdjustmentListener, AncestorListener, AWTEventListener, BeanContextMembershipListener, BeanContextServiceRevokedListener, BeanContextServices, BeanContextServicesListener, CaretListener, CellEditorListener, ChangeListener, ComponentListener, ConnectionEventListener, ContainerListener, ControllerEventListener, DocumentListener, DragGestureListener, DragSourceListener, DragSourceMotionListener, DropTargetListener, FlavorListener, FocusListener, HandshakeCompletedListener, HierarchyBoundsListener, HierarchyListener, HyperlinkListener, IIOReadProgressListener, IIOReadUpdateListener, IIOReadWarningListener, IIOWriteProgressListener, IIOWriteWarningListener, InputMethodListener, InternalFrameListener, ItemListener, KeyListener, LineListener, ListDataListener, ListSelectionListener, MenuDragMouseListener, MenuKeyListener, MenuListener, MetaEventListener, MouseInputListener, MouseListener, MouseMotionListener, MouseWheelListener, NamespaceChangeListener, NamingListener, NodeChangeListener, NotificationListener, ObjectChangeListener, PopupMenuListener, PreferenceChangeListener, PropertyChangeListener, RowSetListener, SSLSessionBindingListener, TableColumnModelListener, TableModelListener, TextListener, TreeExpansionListener, TreeModelListener, TreeSelectionListener, TreeWillExpandListener, UndoableEditListener, UnsolicitedNotificationListener, VetoableChangeListener, WindowFocusListener, WindowListener, WindowStateListener*
- ***All Known Implementing Classes:***
  - *AbstractAction, AbstractButton.ButtonChangeListener, AWTEventListenerProxy, AWTEventMulticaster, BasicButtonListener, BasicColorChooserUI.PropertyHandler, BasicComboBoxEditor, BasicComboBoxEditor.UIResource, BasicComboBoxUI.FocusHandler, BasicComboBoxUI.ItemHandler, BasicComboBoxUI.KeyHandler, BasicComboBoxUI.ListDataHandler, BasicComboBoxUI.PropertyChangeHandler, BasicComboPopup.InvocationKeyHandler, BasicComboPopup.InvocationMouseHandler, BasicComboPopup.InvocationMouseMotionHandler, BasicComboPopup.ItemHandler, BasicComboPopup.ListDataHandler, BasicComboPopup.ListMouseHandler, BasicComboPopup.ListMouseMotionHandler, BasicComboPopup.ListSelectionHandler, BasicComboPopup.PropertyChangeHandler, BasicDesktopIconUI.MouseInputHandler, BasicDesktopPaneUI.CloseAction, BasicDesktopPaneUI.MaximizeAction, BasicDesktopPaneUI.MinimizeAction, BasicDesktopPaneUI.NavigateAction, BasicDesktopPaneUI.OpenAction, BasicDirectoryModel, BasicFileChooserUI.ApproveSelectionAction, BasicFileChooserUI.CancelSelectionAction, BasicFileChooserUI.ChangeToParentDirectoryAction, BasicFileChooserUI.DoubleClickListener, BasicFileChooserUI.GoHomeAction, BasicFileChooserUI.NewFolderAction, BasicFileChooserUI.SelectionListener, BasicFileChooserUI.UpdateAction, BasicInternalFrameTitlePane.CloseAction, BasicInternalFrameTitlePane.IconifyAction, BasicInternalFrameTitlePane.MaximizeAction, BasicInternalFrameTitlePane.MoveAction, BasicInternalFrameTitlePane.PropertyChangeHandler, BasicInternalFrameTitlePane.RestoreAction, BasicInternalFrameTitlePane.SizeAction, BasicInternalFrameUI.BasicInternalFrameListener, BasicInternalFrameUI.BorderListener, BasicInternalFrameUI.ComponentHandler, BasicInternalFrameUI.GlassPaneDispatcher, BasicInternalFrameUI.InternalFramePropertyChangeListener, BasicLabelUI, BasicListUI.FocusHandler, BasicListUI.ListDataHandler, BasicListUI.ListSelectionHandler, BasicListUI.MouseInputHandler, BasicListUI.PropertyChangeHandler, BasicMenuItemUI.MouseInputHandler, BasicMenuUI.ChangeHandler, BasicMenuUI.MouseInputHandler, BasicOptionPaneUI.ButtonActionListener, BasicOptionPaneUI.PropertyChangeHandler, BasicProgressBarUI.ChangeHandler, BasicRootPaneUI, BasicScrollBarUI.ArrowButtonListener, BasicScrollBarUI.ModelListener, BasicScrollBarUI.PropertyChangeHandler, BasicScrollBarUI.ScrollListener, BasicScrollBarUI.TrackListener, BasicScrollPaneUI.HSBChangeListener, BasicScrollPaneUI.MouseWheelHandler, BasicScrollPaneUI.PropertyChangeHandler, BasicScrollPaneUI.ViewportChangeHandler, BasicScrollPaneUI.VSBChangeListener, BasicSliderUI.ActionScroller, BasicSliderUI.ChangeHandler, BasicSliderUI.ComponentHandler, BasicSliderUI.FocusHandler, BasicSliderUI.PropertyChangeHandler, BasicSliderUI.ScrollListener, BasicSliderUI.TrackListener, BasicSplitPaneDivider, BasicSplitPaneDivider.MouseHandler, BasicSplitPaneUI.FocusHandler, BasicSplitPaneUI.KeyboardDownRightHandler, BasicSplitPaneUI.KeyboardEndHandler, BasicSplitPaneUI.KeyboardHomeHandler, BasicSplitPaneUI.KeyboardResizeToggleHandler, BasicSplitPaneUI.KeyboardUpLeftHandler, BasicSplitPaneUI.PropertyHandler, BasicTabbedPaneUI.FocusHandler, BasicTabbedPaneUI.MouseHandler, BasicTabbedPaneUI.PropertyChangeHandler, BasicTabbedPaneUI.TabSelectionHandler, BasicTableHeaderUI.MouseInputHandler, BasicTableUI.FocusHandler, BasicTableUI.KeyHandler, BasicTableUI.MouseInputHandler, BasicTextUI.BasicCaret, BasicToolBarUI.DockingListener, BasicToolBarUI.FrameListener, BasicToolBarUI.PropertyListener, BasicToolBarUI.ToolBarContListener, BasicToolBarUI.ToolBarFocusListener, BasicTreeUI.CellEditorHandler, BasicTreeUI.ComponentHandler, BasicTreeUI.FocusHandler, BasicTreeUI.KeyHandler, BasicTreeUI.MouseHandler, BasicTreeUI.MouseInputHandler, BasicTreeUI.PropertyChangeHandler, BasicTreeUI.SelectionModelPropertyChangeHandler, BasicTreeUI.TreeCancelEditingAction, BasicTreeUI.TreeExpansionHandler, BasicTreeUI.TreeHomeAction, BasicTreeUI.TreeIncrementAction, BasicTreeUI.TreeModelHandler, BasicTreeUI.TreePageAction, BasicTreeUI.TreeSelectionHandler, BasicTreeUI.TreeToggleAction, BasicTreeUI.TreeTraverseAction, BeanContextChildSupport, BeanContextServicesSupport, BeanContextServicesSupport.BCSSProxyServiceProvider, BeanContextSupport, Checkbox.AccessibleAWTCheckbox, Component.AccessibleAWTComponent.AccessibleAWTComponentHandler, Component.AccessibleAWTComponent.AccessibleAWTFocusHandler, ComponentAdapter, Container.AccessibleAWTContainer.AccessibleContainerHandler, ContainerAdapter, DefaultCaret, DefaultCellEditor.EditorDelegate, DefaultEditorKit.BeepAction, DefaultEditorKit.CopyAction, DefaultEditorKit.CutAction, DefaultEditorKit.DefaultKeyTypedAction, DefaultEditorKit.InsertBreakAction, DefaultEditorKit.InsertContentAction, DefaultEditorKit.InsertTabAction, DefaultEditorKit.PasteAction, DefaultTableColumnModel, DefaultTreeCellEditor, DragSourceAdapter, DragSourceContext, DropTarget, DropTarget.DropTargetAutoScroller, DropTargetAdapter, EventListenerProxy, FocusAdapter, FormView, FormView.MouseEventListener, HierarchyBoundsAdapter, HTMLEditorKit.HTMLTextAction, HTMLEditorKit.InsertHTMLTextAction, HTMLEditorKit.LinkController, InternalFrameAdapter, JCheckBox.AccessibleJCheckBox, JCheckBoxMenuItem.AccessibleJCheckBoxMenuItem, JComboBox, JComponent.AccessibleJComponent.AccessibleContainerHandler, JComponent.AccessibleJComponent.AccessibleFocusHandler, JEditorPane.AccessibleJEditorPane, JEditorPane.AccessibleJEditorPaneHTML, JEditorPane.JEditorPaneAccessibleHypertextSupport, JList.AccessibleJList, JMenu.AccessibleJMenu, JMenu.WinListener, JMenuItem.AccessibleJMenuItem, JPasswordField.AccessibleJPasswordField, JPopupMenu.AccessibleJPopupMenu, JRadioButton.AccessibleJRadioButton, JRadioButtonMenuItem.AccessibleJRadioButtonMenuItem, JScrollPane.AccessibleJScrollPane, JSpinner.AccessibleJSpinner, JSpinner.DateEditor, JSpinner.DefaultEditor, JSpinner.ListEditor, JSpinner.NumberEditor, JTabbedPane.AccessibleJTabbedPane, JTabbedPane.ModelListener, JTable, JTable.AccessibleJTable, JTableHeader, JTextArea.AccessibleJTextArea, JTextComponent.AccessibleJTextComponent, JTextField.AccessibleJTextField, JToggleButton.AccessibleJToggleButton, JTree.AccessibleJTree, JTree.TreeModelHandler, JTree.TreeSelectionRedirector, JViewport.ViewListener, KeyAdapter, List.AccessibleAWTList, MetalComboBoxEditor, MetalComboBoxEditor.UIResource, MetalComboBoxUI.MetalPropertyChangeListener, MetalFileChooserUI.DirectoryComboBoxAction, MetalFileChooserUI.FilterComboBoxModel, MetalFileChooserUI.SingleClickListener, MetalLabelUI, MetalRootPaneUI, MetalSliderUI.MetalPropertyListener, MetalToolBarUI.MetalContainerListener, MetalToolBarUI.MetalDockingListener, MetalToolBarUI.MetalRolloverListener, MouseAdapter, MouseDragGestureRecognizer, MouseInputAdapter, MouseMotionAdapter, ProgressMonitor.AccessibleProgressMonitor, PropertyChangeListenerProxy, RelationService, StyledEditorKit.AlignmentAction, StyledEditorKit.BoldAction, StyledEditorKit.FontFamilyAction, StyledEditorKit.FontSizeAction, StyledEditorKit.ForegroundAction, StyledEditorKit.ItalicAction, StyledEditorKit.StyledTextAction, StyledEditorKit.UnderlineAction, TextAction, TextArea.AccessibleAWTTextArea, TextComponent.AccessibleAWTTextComponent, TextField.AccessibleAWTTextField, ToolTipManager, ToolTipManager.insideTimerAction, ToolTipManager.outsideTimerAction,*
- *public interface* ***EventListener***
  - *A tagging interface that all event listener interfaces must extend.*"

# Was ist an diesem Beispiel schlecht?



- Offenbar ist das eine recht wichtige Schnittstelle
  - jedenfalls wird sie verdammt häufig benutzt

- Aber es bleiben zentrale Fragen offen:
  - Was macht denn überhaupt einen event listener aus?
  - Wer profitiert davon, wenn ich diese Schnittstelle implementiere?
  - Was bedeutet "must implement"?
  - Was passiert, wenn ich es nicht tue?
  - Oder was passiert nicht?

# Arten von Dokumentation

**Externe Dokumentation** (für Anwender):

- Benutzeranleitungen
  - Lernanleitungen, Bedienhandbücher, Referenzhandbücher
- Installationsanleitungen
- Administrationsanleitungen
- Versionsbeschreibung
  - "release notes", Änderungen zur Vorversion, bekannte Mängel, etc.

**Interne Dokumentation** (für Entwickler):

- Produktdokumentation:
  - ***Anforderungsbeschreibgn.***
  - ***Entwurfsbeschreibungen***
    - ***Architektur***
    - ***Module***
    - ***Entwurfsentscheidungen und -begründung***
  - Details zur Implementierung
  - Testfallbeschreibungen
  - Beziehungen zwischen diesen
  - etc.
- Prozessdokumentation:
  - Projektplan u. Teilpläne
  - Testhistorie
  - Änderungshistorie
  - etc.

# Dokumentation von Anforderungen

Freie Universität Berlin

Struktur eines Anforderungsdokuments: [s. Vl. "Dyn. Modellierung"]

1. Introduction
2. Current system
3. Proposed system
   - 3.1 Overview
   - 3.2 Functional requirements [siehe Vorlesung "Use Cases"]
   - 3.3 Nonfunctional requirements [s. Vl. "Dyn. Modellierung"]
   - 3.4 Constraints ("Pseudo requirements") [dito]
   - 3.5 System models
     - 3.5.1 Scenarios [siehe Vorlesung "Use Cases"]
     - 3.5.2 Use case model [siehe Vorlesung "Use Cases"]
     - 3.5.3 Object model [s. Vl. "Statische Modellierung"]
     - 3.5.4 Dynamic models [s. Vl. "Dyn. Modellierung"]
     - 3.5.5 User interface
4. Glossary

# Dokumentation von Architekturentwurf

Freie Universität Berlin

Besteht aus zwei Teilen:

- Dokumentation des Grobentwurfs
  - Subsysteme und ihre Schnittstellen
  - z.B. durch UML-Komponentendiagramme plus OCL-Spezifikation oder textuelle Beschreibung
- Beschreibung wie/warum damit die nichtfunktionalen Anforderungen erreicht werden
  - In der Regel Text
    - Klare Bezüge auf das Anforderungsdokument
  - Evtl. quantitative Argumentation (z.B. bei Leistungsanforderungen) oder Beweisführungen (z.B. bei Sicherheitsanforderungen)
- Beide Teile können sehr dadurch vereinfacht werden, dass man sich auf eine bewährte (Standard)Architektur bezieht

# Dokumentation von Modulentwurf



"Modulführer" (realisiert z.B. als javadoc o.ä.):

- Beschreibt für jedes Modul
  - Importschnittstelle, d.h. welche anderen Module es benutzt
  - Exportschnittstelle, d.h. welche Datentypen und Dienste es anbietet und welches deren genaue Bedeutung ist
- Für jeden Dienst/Operation/Methode:
  - Voraussetzungen (precondition, "Vorbedingung")
  - Effekt (Wirkung, postcondition, "Nachbedingung")
  - **Zweck** (für Aufrufer) und **Entwurfsbegründung** (für Wartung)
- Voraussetzung und Effekt kann man z.B. mit OCL notieren
  - muss aber nicht sein und geht auch oft zu schwierig
- Zweck ist ebenso wichtig und kann nur mit Text beschrieben werden
  - Manchmal sind Voraussetzung und Effekt offensichtlich, der Zweck jedoch nicht

# Rationale management ("Begründungs-mgmt."): An aircraft example

Freie Universität Berlin

A320 (first flight in 1988)

- First fly-by-wire passenger aircraft
- 150 seats, short to medium haul


A318 (2003), A319 (1996), A321 (1994)

- Derivatives of A320
- Same handling as A320

Design rationale

- Reduce pilot training & maintenance costs
- Increase flexibility for airline

# An aircraft example (2)



A330 (1993) & A340 (1993)

- Long haul and ultra long haul
- 2x seats, 3x range
- Similar handling than A320 family

Design rationale

- With minimum cross training, A320 pilots can be certified to fly A330 and A340 airplanes

Consequence

- Any change in these five airplanes must maintain this similarity

# What is rationale?



- Rationale is the reasoning underlying the system

- Rationale includes:
  - the **issues** that were addressed
    - Themen, Fragestellungen
  - the **alternatives** that were considered
    - Entwurfsmöglichkeiten
  - the **decisions** that were made to resolve the issues
    - Entwurfsentscheidungen
  - the **criteria** that were used to guide decisions, and
    - Gewichtungs- und Auswahlkriterien (Qualitätsmaßstäbe)
  - the **arguments** developers went through to reach a decision
    - Auswahlbegründungen

# Why is rationale important in software engineering?

Freie Universität Berlin

Many software systems are like aircraft:

- They result from a large number of decisions taken over an extended period of time

- Evolving assumptions
- Legacy decisions
- Conflicting criteria

→ high maintenance cost

→ loss & rediscovery of information

# Uses of rationale in software engineering

Freie Universität Berlin

- Improve maintenance support
  - Provide maintainers with design context, so they can avoid violating the design ideas
- Improve design support
  - Avoid duplicate evaluation of poor alternatives
  - Make consistent and explicit trade-offs
- Improve documentation
  - Makes it easier for non-developers (e.g., managers, lawyers, technical writers) to review the design
- Improve learning
  - New staff can learn the design by replaying the decisions that produced it

# Representing rationale: issue models

Freie Universität Berlin

- Argumentation is the most promising approach so far:
  - Provides more information than a design document: captures trade-offs and discarded alternatives that design documents do not
  - Is less messy than communication records: communication records contain everything, are redundant and ill-structured

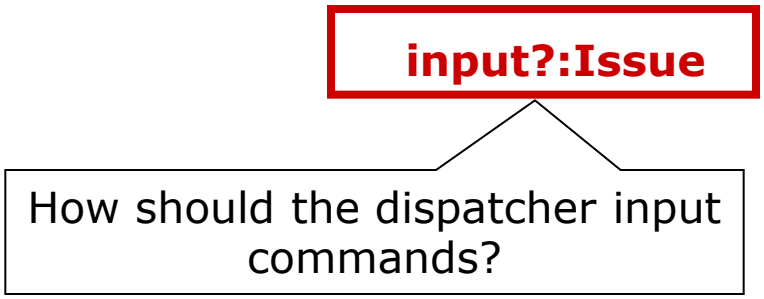
- Issue models represent arguments in a semi-structured form:
  - Nodes represent argument steps
  - Links represent their relationships

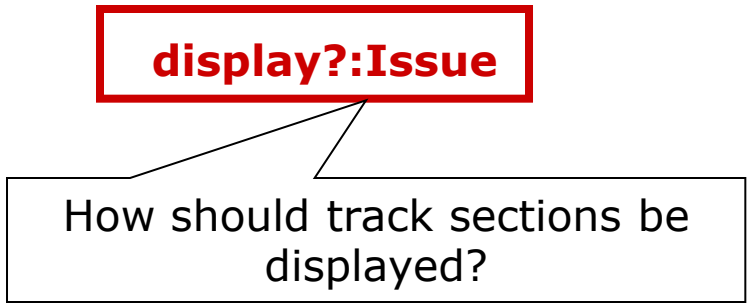
# Rationale description concepts: Issues



- Issues are concrete problems which usually do not have a unique, correct solution
- Issues are phrased as questions

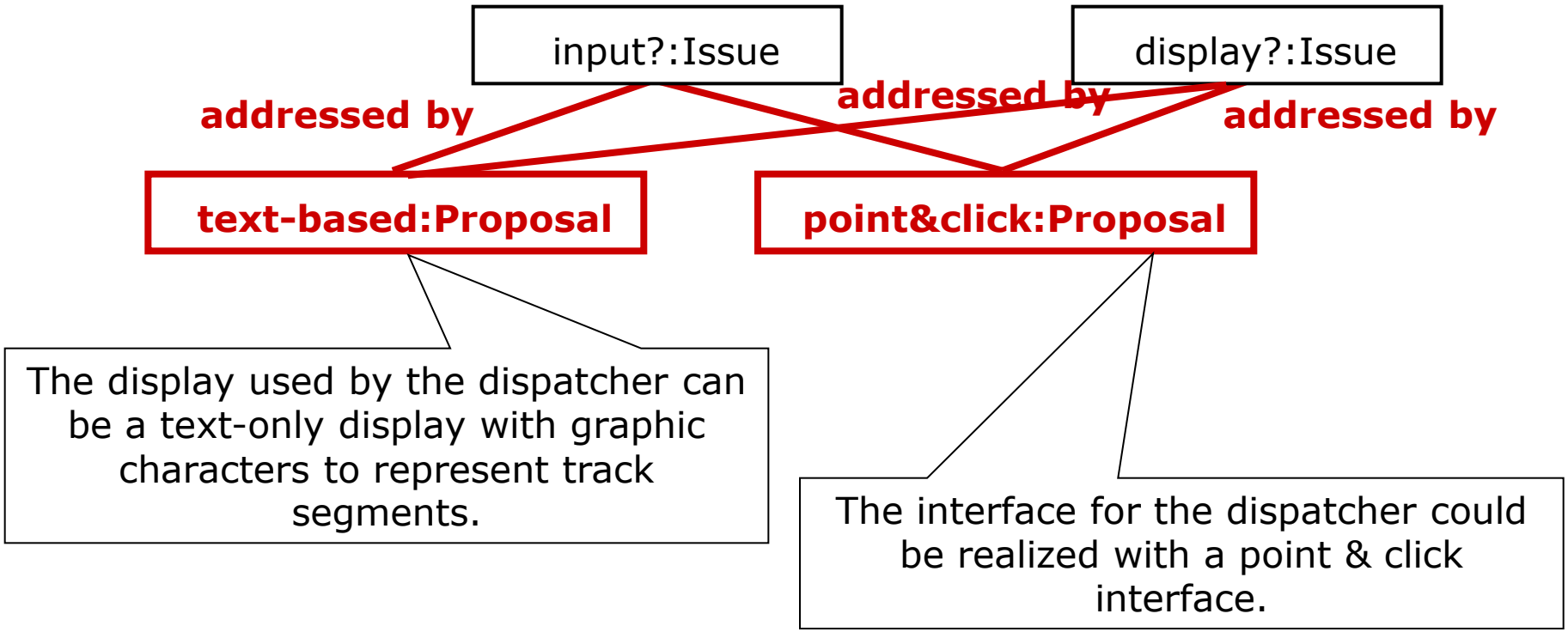
Example domain:
A railway traffic mgmt system



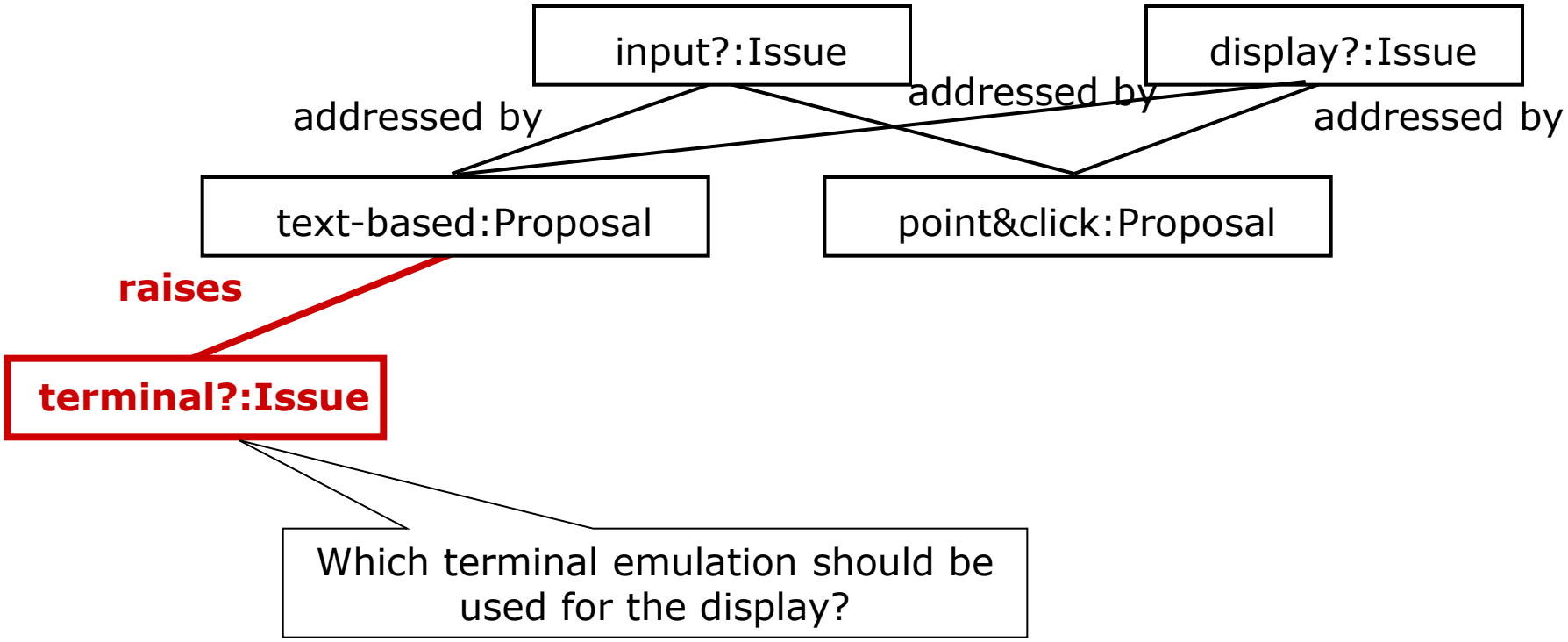
# Rationale description concepts: Proposals

- Proposals are possible alternative solutions for issues
- One proposal can be shared across multiple issues



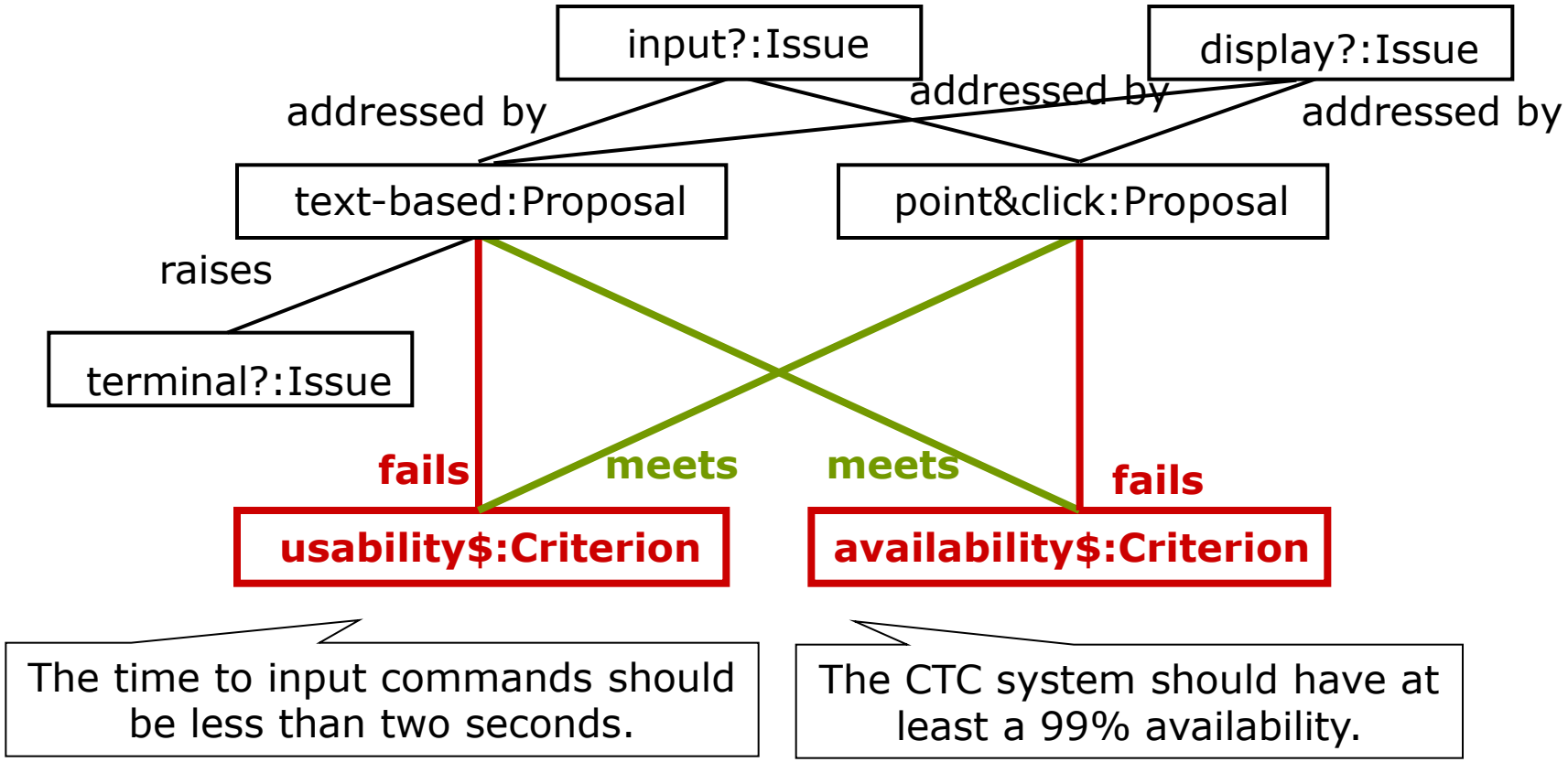
# Rationale description concepts: Consequent issue

- Consequent issues are issues raised by the introduction of a proposal



# Rationale description concepts: Criteria

- A criterion represents a measure of goodness
- Criteria are often design goals or nonfunctional requirements



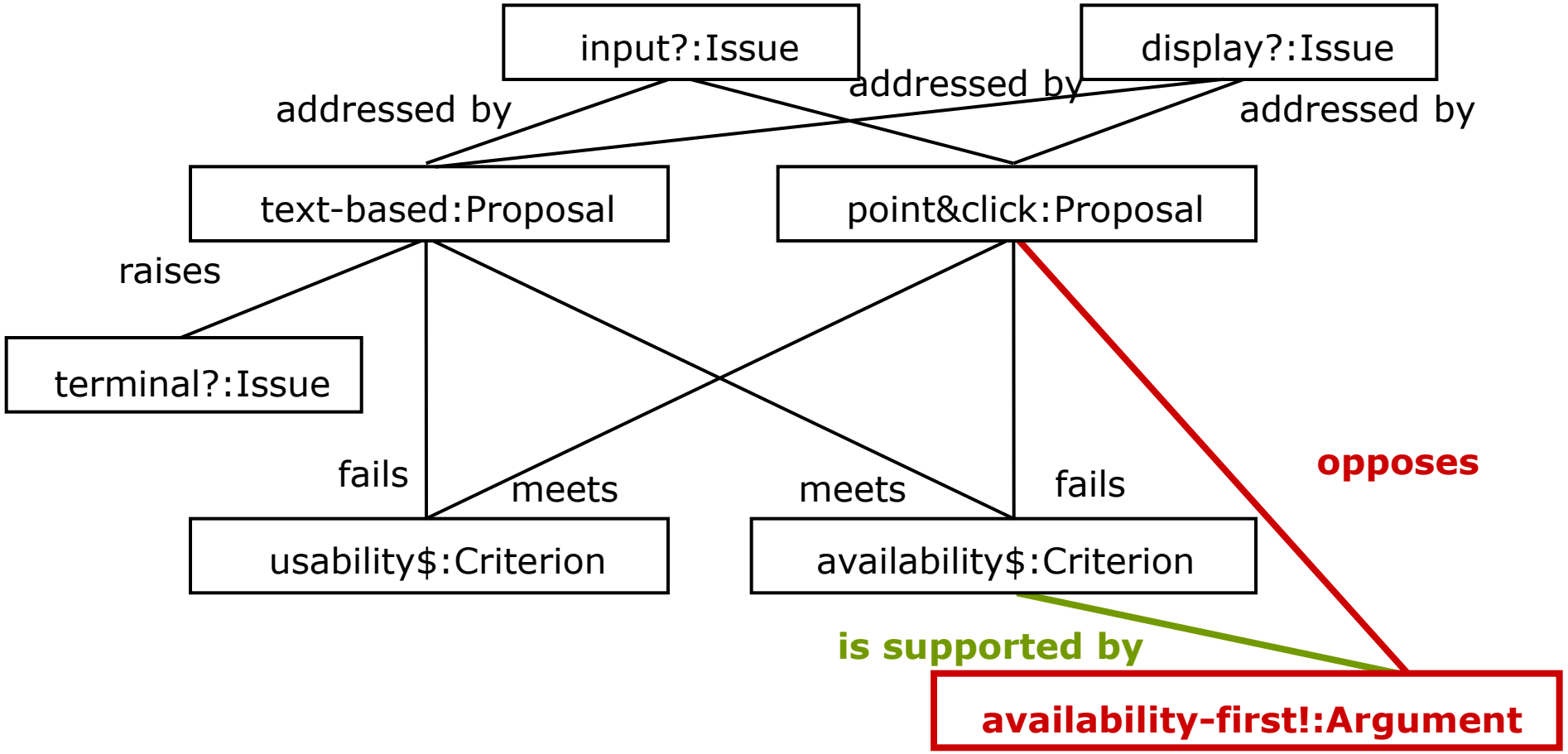
# Rationale description concepts: Arguments

Freie Universität Berlin

- Arguments represent the debate developers went through to resolve the issue
- Arguments can also support or oppose any other part of the rationale
- Arguments constitute the main part of rationale

# Rationale description concepts: Arguments (2)

Freie Universität Berlin



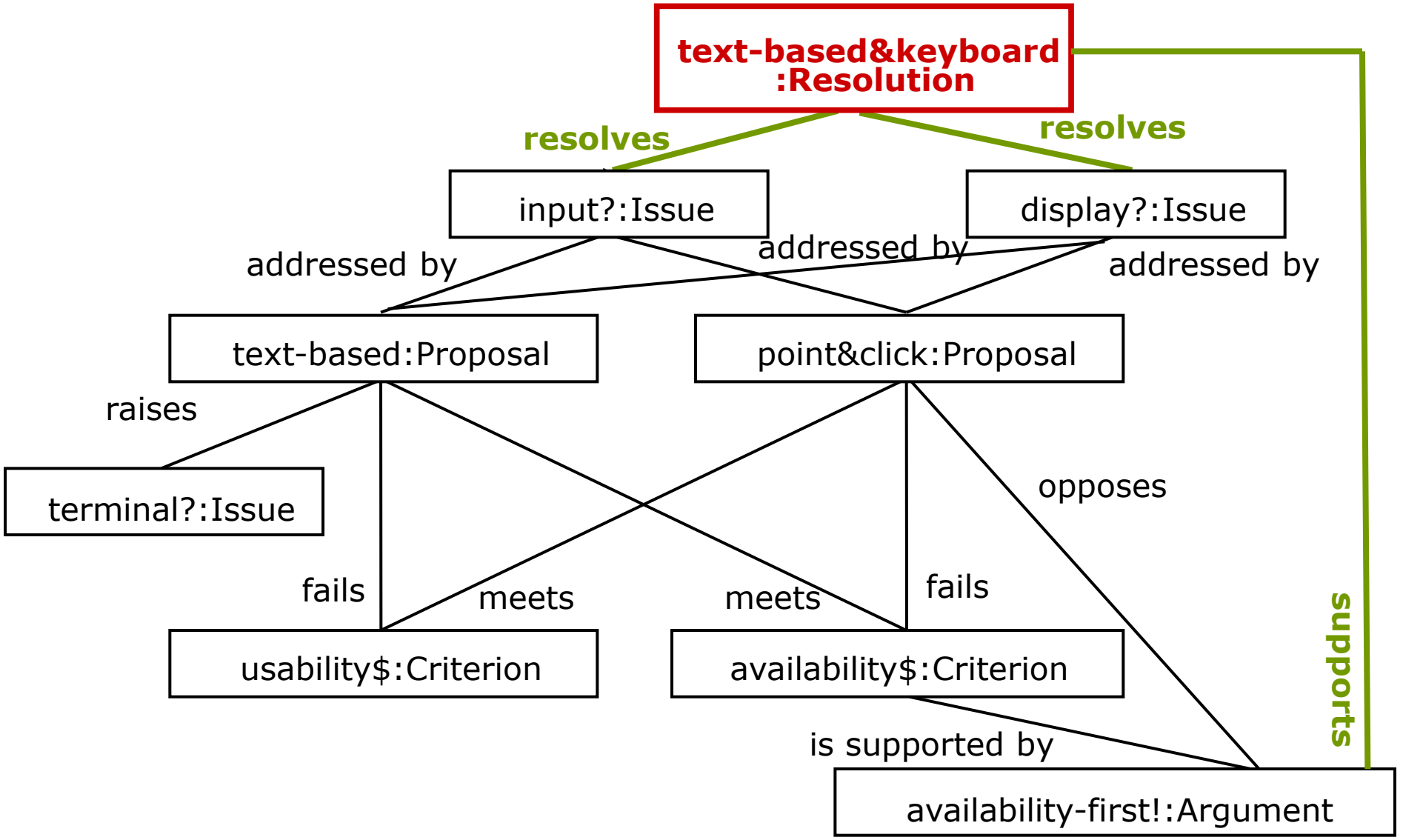

Point&click interfaces are more complex to implement than text-based interfaces and more difficult to test.
The point&click interface risks introducing fatal errors in the system that would offset any usability benefit.

# Rationale description concepts: Resolutions

Freie Universität Berlin

- Resolutions represent decisions
- A resolution summarizes the chosen alternative and the argument supporting it
- A resolved issue is said to be closed
  - but can be re-opened if necessary

# Rationale description concepts: Resolutions (2)



# And now for something completely different...

# Einige typische Englisch-Fehler deutscher Muttersprachler

Freie Universität Berlin

Wortfehler (false friends):

- "der aktuelle Wert"
  - "the *current* value"
  - "actual": "tatsächlich, wirklich"
- "der String bekommt eine Längenangabe"
  - "a length indicator is added to the String"
  - "becomes": "wird zu"
- "mein Chef"
  - "my *boss*", "my *manager*"
  - "chef": "Koch, Chefkoch"
- "Handy": "mobile phone"
- "Sitzungsprotokoll": "minutes"
- "ein ernster Fehler"
  - "a *serious* error"
- "ein seriöses Angebot"
  - "a *trustworthy* offer"
- "ein eigener Fall"
  - "a *separate* case", "a case of its own"
- "eventuell läuft der Zähler über"
  - "the counter *may* overflow"
  - "eventually": "schlussendlich"
- "eine Milliarde" (giga)
  - "one *billion*" (giga)
- "eine Billion" (tera)
  - "one *trillion*" (tera)

# Englisch-Fehler (2)

Pluralfehler:

- "zwei Kinder"
  - "two child*ren*"
- "die Informationen"
  - "the *information*"
- "die Personen im Raum"
  - "the *people* in the room"
  - "persons" gibt es, klingt aber selten richtig
- "die Daten sind zerstört"
  - "the data *are* corrupt"

Zeichensetzung:

- "A, B und C"
  - "A, B, and C"
    - "serial comma"
  - machen auch Amis meist falsch

Ausdrucksweisen:

- Endloses Thema. Beispiel:
  - "Es gibt eine Lösung" →
  - "It gives a solution": falsch
  - "It exists a solution": furchtbar
  - "There *is* a solution": richtig
  - "Äpfel mit Birnen vergleichen"
  - → "Comparing apples and *oranges*"
  - usw. usf.

Resources for ESL students

- "English as a second language"
- http://www.manythings.org/
- http://a4esl.org/
- http://www.eslgold.com/
- http://uebersetzungsfallen.de/

# Zusammenfassung



- Es gibt zahlreiche Arten von Dokumentation
  - einzuteilen in interne und externe Dokumentation
- Gute Dokumentation ist übersichtlich, präzise, korrekt und vor allem **hilfreich**
  - Sie zu schreiben ist ein Handwerk: üben, üben, üben
  - Seien Sie selbstkritisch!
- Begründungsmanagement dokumentiert Entwurfsentscheidungen
  - Wie aus Fragestellungen (issues) und Lösungsmöglichkeiten (proposals) mit Hilfe von Kriterien und Argumenten eine Entscheidung (resolution) ausgewählt wurde
- Lernen Sie passables Englisch!

[Meine] Frau, die ja tatsächlich immer noch Bücher liest, kommt an und fragt:
"Was ist eine Landmarke?"
Ich sage: "Das ist Englisch und steht unter *landmark* im Lexikon."
"Und was finde ich, wenn ich *landmark* nachschlage?"
"Wahrzeichen"
"Und warum schlägt der Übersetzer nicht selbst nach?"
"Hm."

[Harry Rowohlt: Pooh's Corner, DIE ZEIT 17.07.2008]

# Danke!