



A Coding Scheme for Actual Programming Processes

Sebastian Jekutsch
Institut für Informatik
FU Berlin

Contents

1. Introduction
2. General Model of Representation
 - incl. examples
3. Model, Schema, Actual Process
4. Stephan's Model
5. Visualization
 - incl. example

You are here



- Actual process research: Study of what really happens during software development
 - Here: Restricted to programming = coding and related activities
- I will present a *Coding* Scheme = Language to describe what's going on
 - Interpreting videos or basic technical actions
- Aim is to...
 - provide an abstraction for reasoning
 - develop helpful measures
 - allow for discovery of behavior patterns
 - explore possible reasons for making errors

Main Example (1)

Video:

```

ConstraintResultList res=riter(r);
    return res.getOutput().getResourcePairList().iterator();
}

protected static String[] emptyStringArray=new String[] {};
public String[] getDetailsKeys() {
    return emptyStringArray;
}
public String[] getDetailsDescriptions(ResourcePair r) {
    return emptyStringArray;
}
    
```

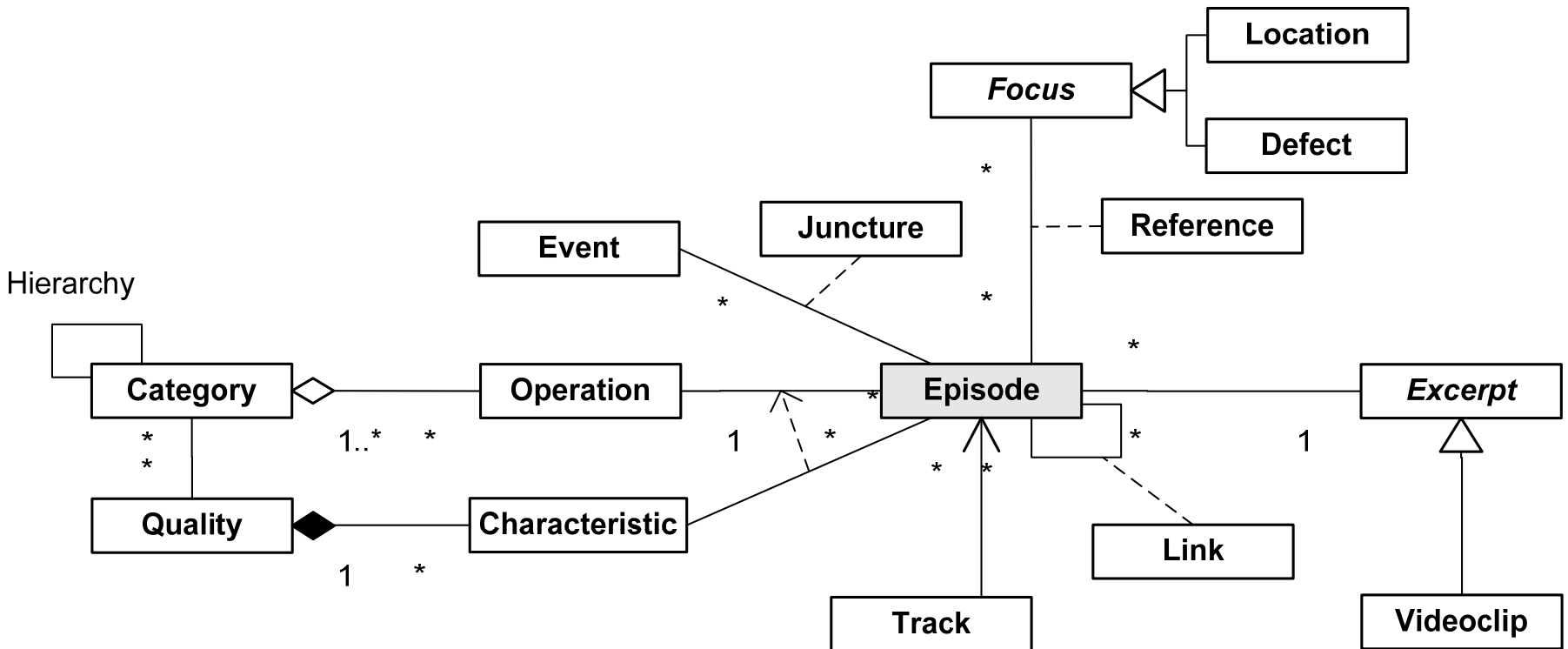
Basic technical actions:

cursor move, change, change, .. copy, paste, change, .. warning

Interpretation:

Programmer adds a new method called „getDetailsDescriptions“ to class „Base“ via copying previously written code and altering it afterwards. A syntactical defect has been introduced.

Generic model of representation

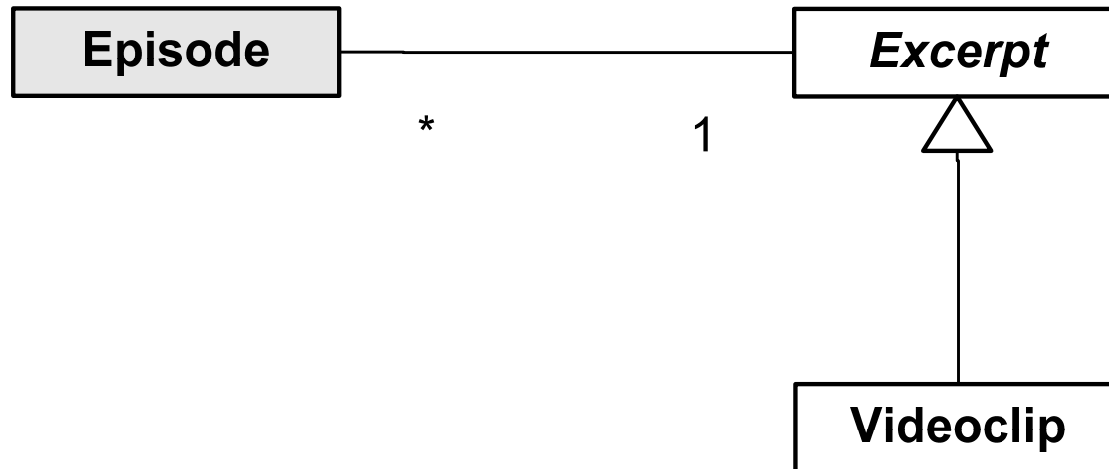


Central coding concept



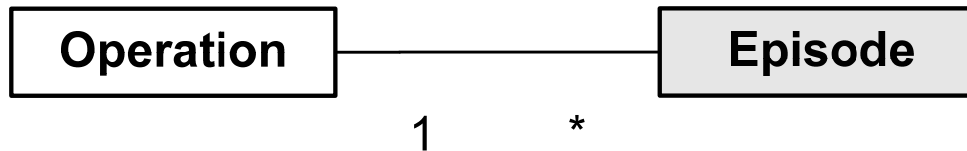
Excerpt (Ausschnitt)

- Excerpt = time frame during development where “one operation” happens
 - video clip excerpts usually contain more than one operation
- Longest possible sequence without any relevant change of situation/status
 - no influencing events, no mental state change, no new action, etc.

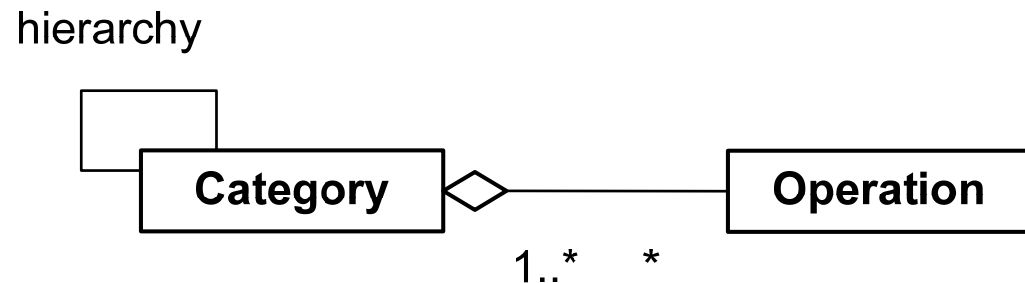


Operation (Vorgang)

- Mostly activities of the programmer
 - includes mental processes and non-actions



- Operations are classified in Categories (Kategorien)
 - which form a hierarchy
- Current high level Operation Categories:
 - Activities
 - Phases



Category „Activities“: Overview

- Core activities
 - Changing code
 - Browsing
 - Reading, Thinking, Pausing
 - Other kind of work
- Inner activities
 - Changing code sideways
 - Eye (=Cursor) movements
- Batch activities
 - Compiling
 - Executing program / Testing

Core activities

- Changing Code:
 - Advancement, Adjustment, Betterment, Complement, Displacement, Document, Embellishment
- Browsing:
 - ForOperator, ForSignature, ForLocation, ForClass
- Reading:
 - Requirements, Reorientation, Reviewing
- Thinking:
 - AboutWhatNextToDo, AboutEvent, AboutProblem
- Pausing:
 - OtherInterest, Waiting, Indifferent
- Other kind of Working:
 - Delivering, ExplainingToOthers, PreparingData, UsingEnvironment, UsingOS, UsingProgram

Core activities, Examples (1)

- ChangingCode.Advancement -> Main Example
- ChangingCode.Complement:

```

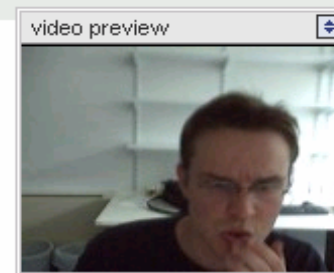
public void fstPass(Vector Words)
{
    for (int i=0; i<Words.size(); i++)
    {
        for (int j=i; j<Words.size(); j++)
        {
            String w1 = (String) Words.elementAt(i);
            String w2 = (String) Words.elementAt(j);

            if (w1.indexOf(w2) != -1)
                Words.remove(w2);
        }
    }
}

public void sndPass(Vector Words)
{
    for (int i=0; i<Words.size(); i++)
    {
        for (int j=i; j<Words.size(); j++)
        {
            String w1 = (String) Words.elementAt(i);
            String w2 = (String) Words.elementAt(j);

            if (w1.indexOf(w2) != -1)
                Words.remove(w2);
        }
    }
}

```



Core activities, Examples (2)

- ChangingCode.Betterment:

```

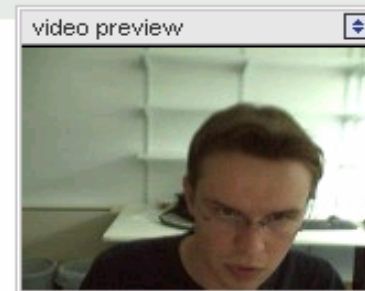
public void fstPass(Vector Words)
{
    for (int i=0; i<Words.size(); i++)
    {
        boolean bContinue=true;

        while (bContinue)
        {
            for (int j=i; j<Words.size(); j++)
            {
                String w1 = (String) Words.elementAt(i);
                String w2 = (String) Words.elementAt(j);

                if (w1.indexOf(w2) != -1)
                {
                    Words.remove(w2);
                    bContinue = true;
                    break;
                }
            }
        }
    }
}

public void sndPass(Vector Words)
{
    for (int i=0; i<Words.size(); i++)
    {
        for (int j=i; j<Words.size(); j++)
        {

```



Inner and Batch activities


Inner activities:

- Code Changing by the way
 - Tag-Along, Re-Spell, Work-Over, Dust-Off, Hold-On, Plan-Aloud, Look-Up, Spruce-Up
- Browsing by the way:
 - Cursor Moving, Scrolling, Selecting

Batch activities:

- Compile, Build
- Run, Debug

Inner activities, Examples

- Re-Spell + Tag-Along: 
- Thinking.AboutProblem includes Plan-Aloud:

```

public void fstPass(Vector Words)
{
    for (int i=0; i<Words.size(); i++)
    {
        boolean bContinue=true;

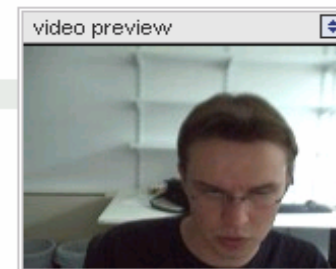
        while (bContinue)
        {
            bContinue=true;
            for (int j=i; j<Words.size(); j++)
            {
                String w1 = (String) Words.elementAt(i);
                String w2 = (String) Words.elementAt(j);

                if (w1.indexOf(w2) != -1)
                {
                    Words.remove(w2);
                    break;
                }
            }

            bContinue = false;
        }
    }
}

public void sndPass(Vector Words)
{
    for (int i=0; i<Words.size(); i++)

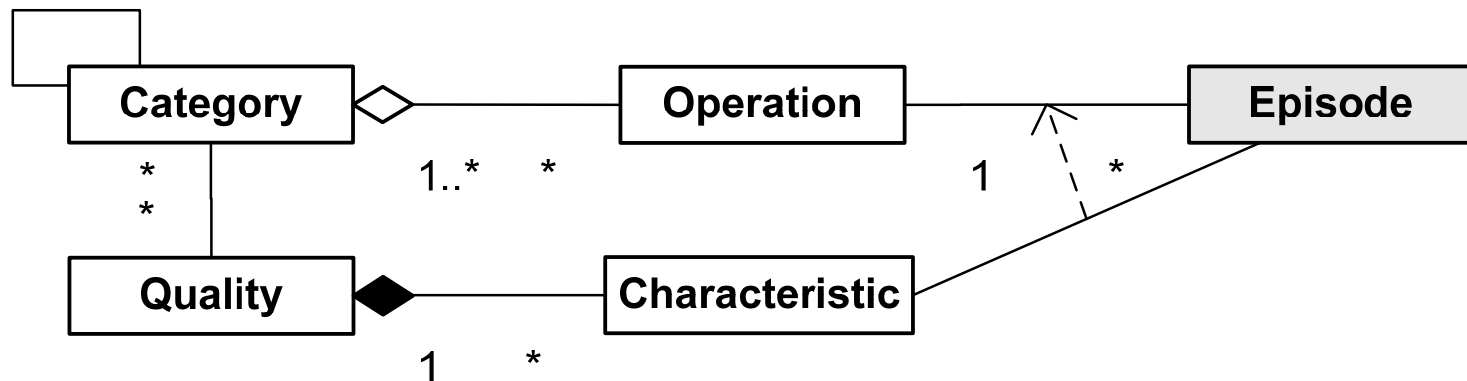
```



Quality (Eigenschaft) and Characteristic (Ausprägung) (1)

- Operations have optional Qualities
 - inherited from Category
 - which itself inherits from its Super-Category
- Episodes have Characteristics
 - based on Operation's Qualities

hierarchy



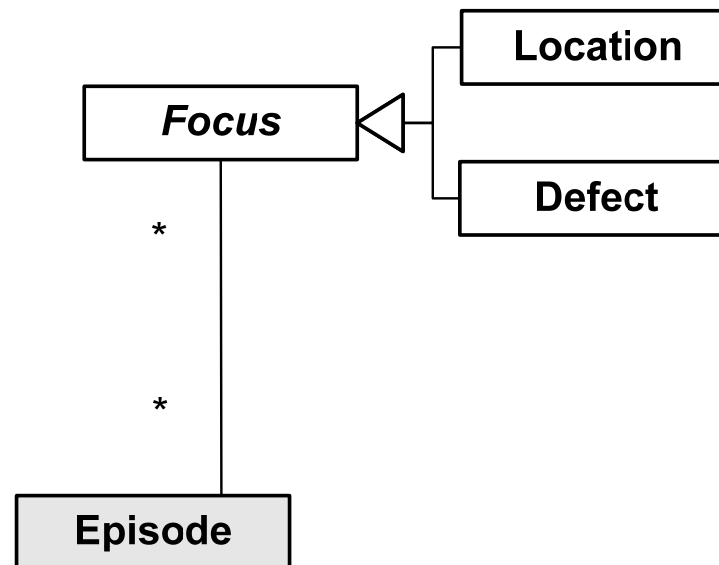
Quality (Eigenschaft) and Characteristic (Ausprägung) (2)

- Examples for Qualities/Characteristics:
 - Source (for Code changes)
 - BrainDump, CopyPaste, TypeWrite
 - Speed (for Core Activities)
 - Slower, Faster
 - Finish (for Code changes)
 - LeavesUncomplete, LeavesOpen
 - Pressure (for Activities)
 - Higher

- *Legend:*
 - Quality (for Category)
 - Characteristics of this Quality

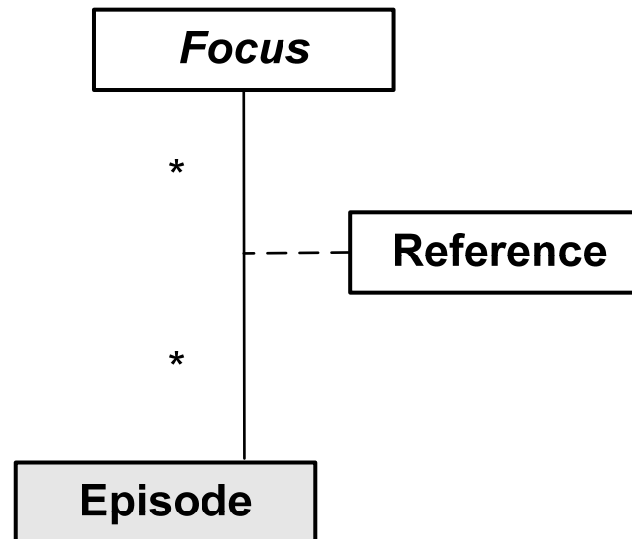
Focus (Fokus)

- Object which the Operation refers to
 - (Code-)Location
 - Defect (semantic, syntactic)
 - others will likely be added
- One Operation may relate to different Foci



Reference (Bezug)

- ... for Locations:
 - creates, changes/corrects, discards, documents/explains, extends, copies
- ... for Defects:
 - introduces, detects, resolves



Focus/Reference/Quality/ Characteristic Example

- ChangingCode.Complement {Speed=Slower} changes fstPass(), creates SemanticDefect#1:

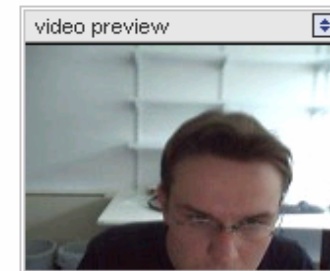
```
public void fstPass(Vector Words)
{

    for (int i=0; i<Words.size(); i++)
    {
        boolean bContinue=true;

        while (bContinue)
        {
            for (int j=i; j<Words.size(); j++)
            {
                String w1 = (String) Words.elementAt(i);
                String w2 = (String) Words.elementAt(j);

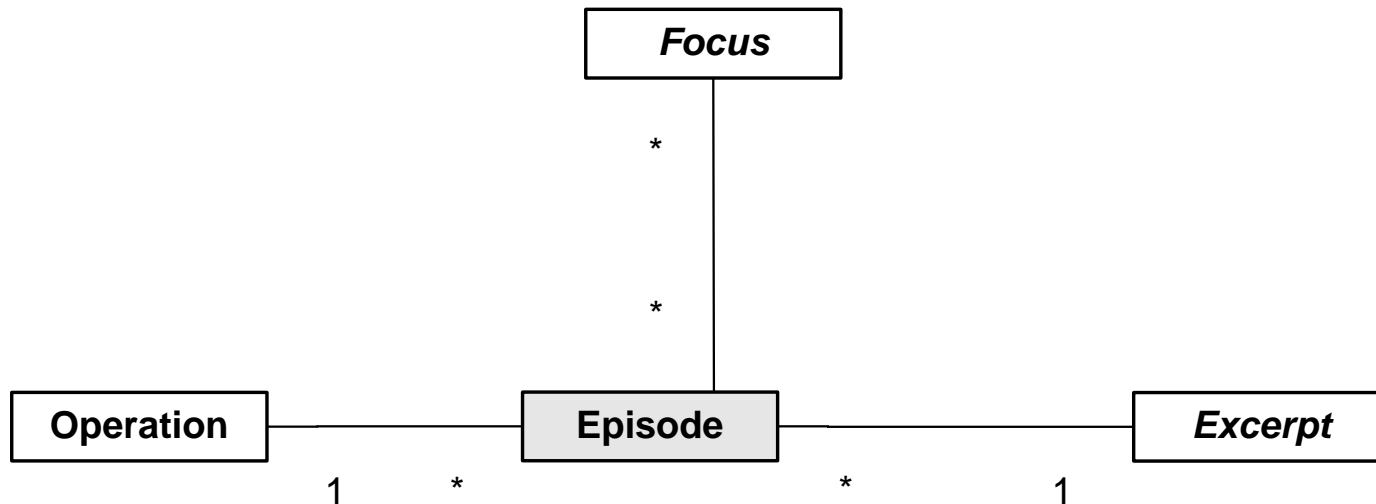
                if (w1.indexOf(w2) != -1)
                {
                    Words.remove(w2);
                    j--;
                }
            }
        }
    }

    public void sndPass(Vector Words)
    {
        for (int i=0; i<Words.size(); i++)
        {
            for (int j=i; j<Words.size(); j++)
```



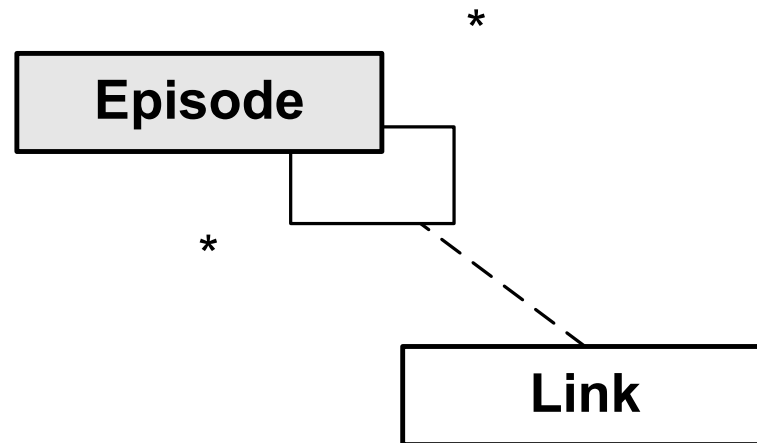
Episode (Episode)

- Connection of Operation and Excerpt
 - with optional Reference to a Focus of Operation
- Should an Excerpt have only one Episode?
 - or: is Excerpt reusable?
- Capturing the actual process = Recognizing Episodes, i.e. Operations in Excerpts



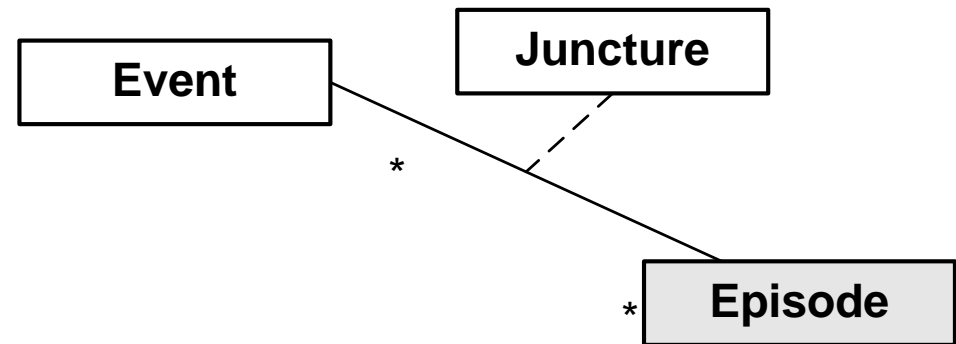
Link (Verknüpfung)

- Operations may resume other Operations
 - telling a story without interruptions and other parallel activities
 - often in Reference to the same Focus
- Currently only:
 - resumes
 - contains?



Event (Ereignis)

- Event = impact on programmer's behavior
 - from outside
 - probably as an effect of her own activity
 - at a point of time
- Examples:
 - Interruptions
 - Result of compilation
 - Result of test



- Juncture (Punkt):
 - „Beginning“, „End“, „In Between“ of an Excerpt

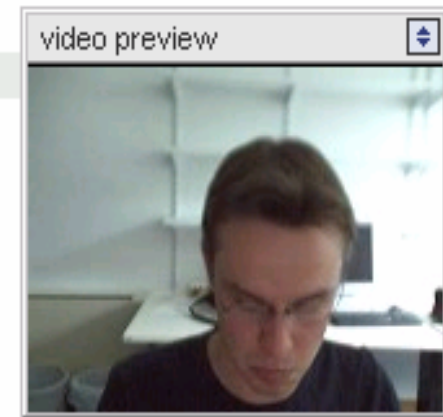
Event Example

- ChangingCode.Advancement {Finish = LeavesOpen} changes sndPass() | Interruption:

```
public void sndPass(Vector Words)
{
    for (int i=0; i<Words.size(); i++)
    {
        if (Words.size()<=i) break;

        for (int j=i; j<Words.size(); j++)
        {
            if (Words.size()<=j) break;

            String w1 = (String) Words.elementAt(i);
            String w2 = (String) Words.elementAt(j);
```



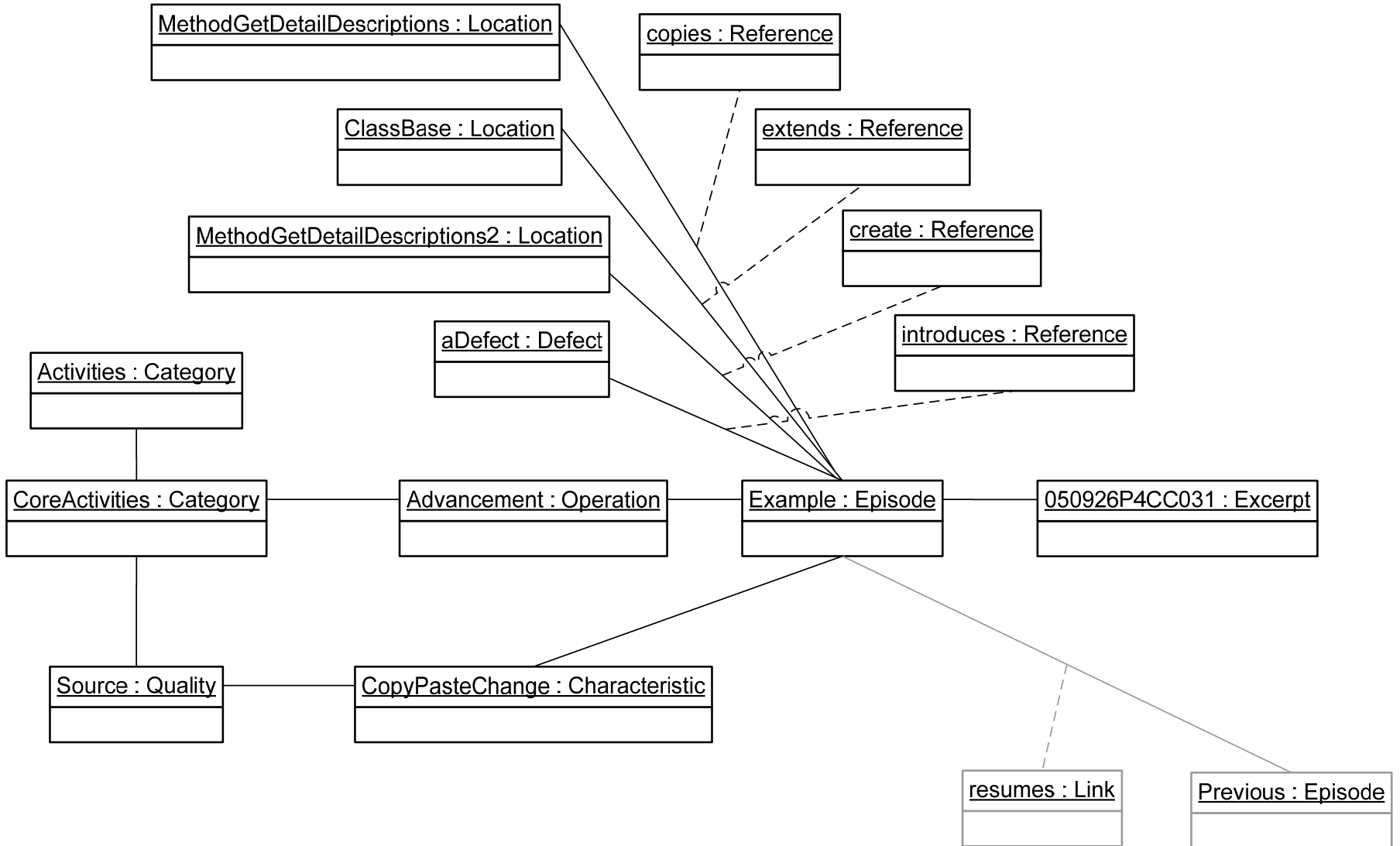
Main Example (2)

- ChangingCode.Advancement {Source = CPC} extends Base, creates getDetailsDescriptions2, copies getDetailsDescriptions, introduces aDefect:

```
        ConstraintResultList res=filter(r);
        return res.getOutput().getResourcePairList().iterator();
    }

    protected static String[] emptyStringArray=new String[] {};
    public String[] getDetailsKeys() {
        return emptyStringArray;
    }
    public String[] getDetailsDescriptions(ResourcePair r) {
        return emptyStringArray;
    }
}
```


Main Example (3)



Contents

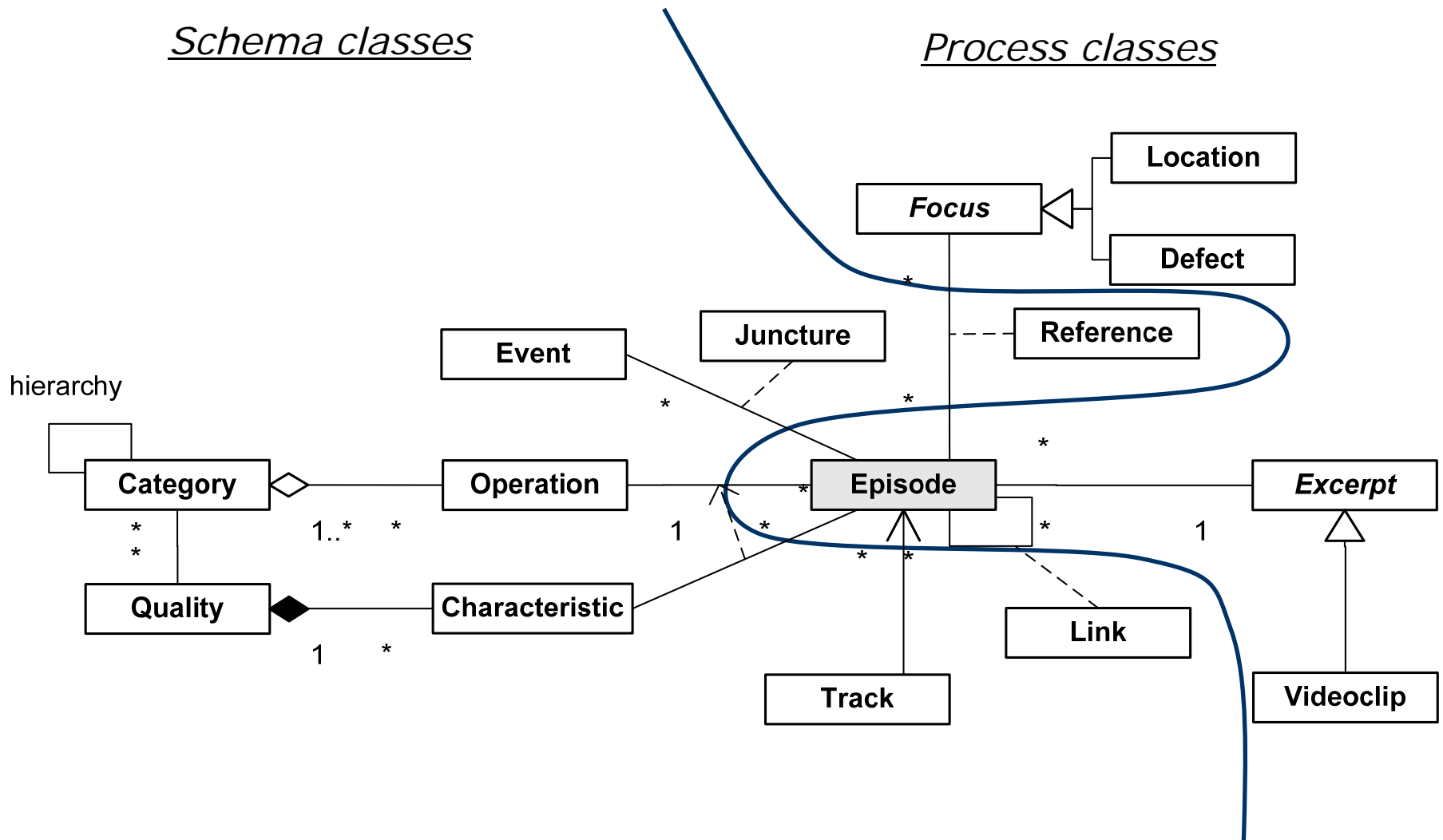
1. Introduction
2. General Model of Representation
 - incl. examples
3. Model, Schema, Actual Process
4. Stephan's Model
5. Visualization
 - incl. example

You are here

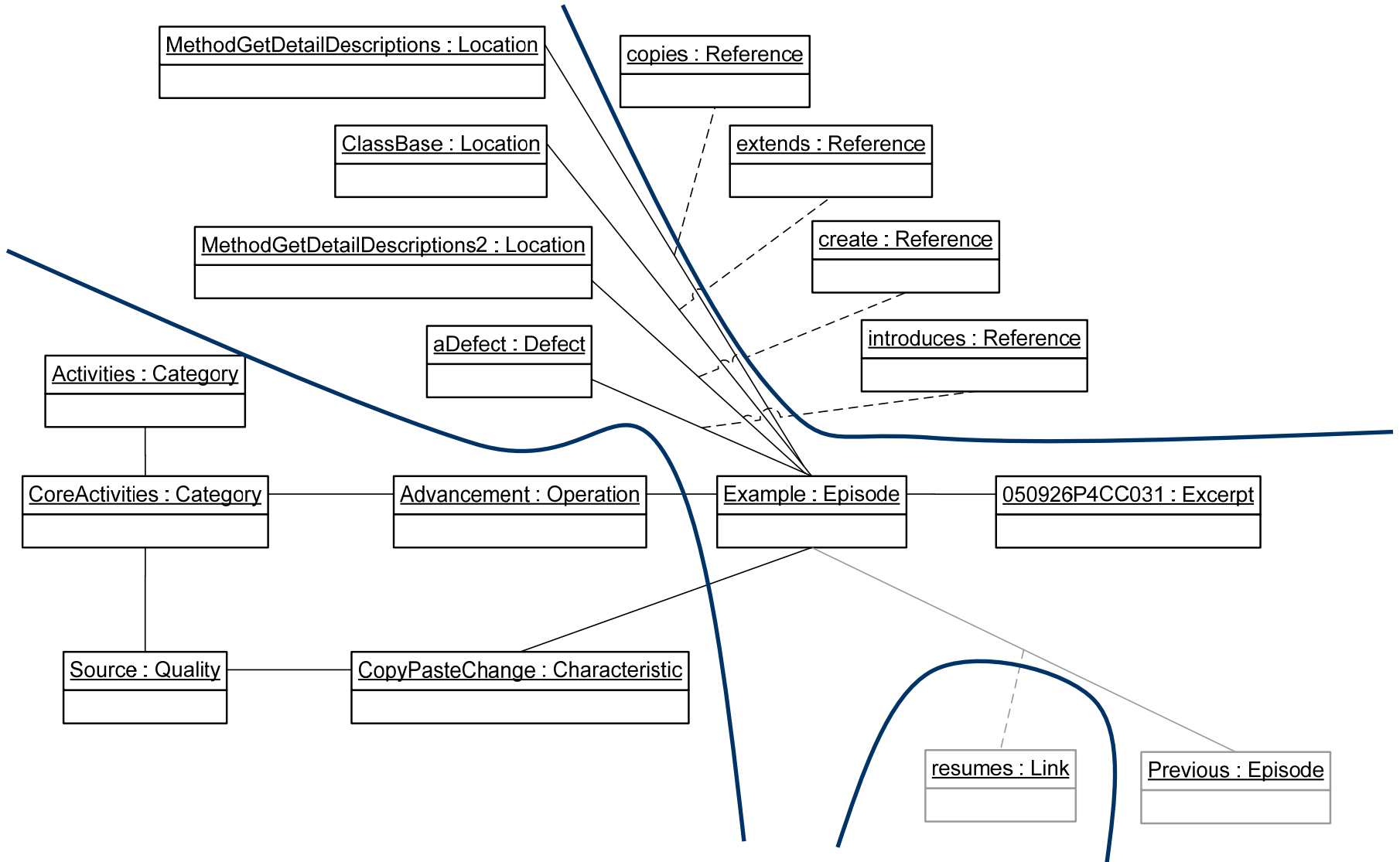


Model, Schema, Actual processes

- Model: Contains general, constant concepts (grammar) about work episodes
- Schema: Contains potentially existent concepts for describing programming (vocabulary)
 - Objects of classes Operation, Event, Category, Property, Characteristic, Reference, Link
 - and Associations between them
 - will be extended with further insight
- Actual process: Contains concepts of an observed programming session (sentences)
 - Objects of classes Episode, Focus, Excerpt
 - and its associations among each other, Events and Operations




Example, revisited



Capturing the actual process

1. On the lowest level, actual processes are streams of basic technical events
 - File change, clicking Run, etc.
 2. Group events to excerpts = time frames
 - Events are not necessarily in one consecutive row
 - But still without status changing during the episode
 3. Assign Excerpt an Interpretation as an Operation
 - leading to an Episode
 4. Add Foci and Links
- “Process Disassembling”: It’s like reconstructing the chicken out of the chicken soup!
 - Can it be done automatically?

Comparison with Stephan's model

- Episode = Annotation
- Operation \leq Concept, Category = Concept Class
- Excerpts may contain different Concepts
- Properties, Characteristics and Events are Concepts
 - of Classes "Properties", "Characteristics" and "Event"
- Foci (Locations, Defects) are Excerpts as well 
 - which do have a range, but not necessarily in time
 - Links and References become synonyms
- Multiplicities of associations change accordingly
- Stephan's model contains no concept class hierarchy
 - but Inheritance of Annotations, Primary annotations
 - and Concepts in more than one Classes
- Stephan's model is more suitable for manual annotation

Contents

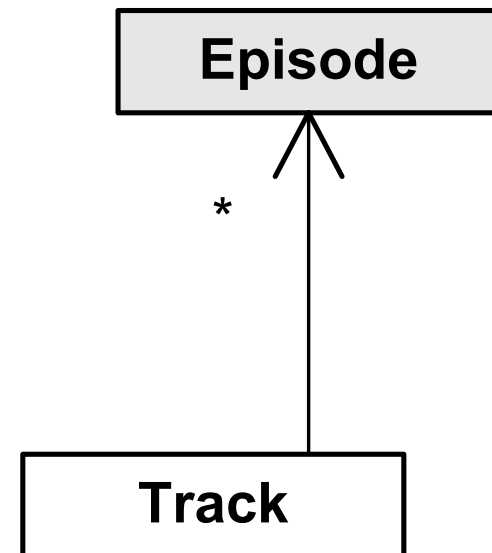
1. Introduction
2. General Model of Representation
 - incl. examples
3. Model, Schema, Actual Process
4. Stephan's Model
5. Visualization
 - incl. example

You are here



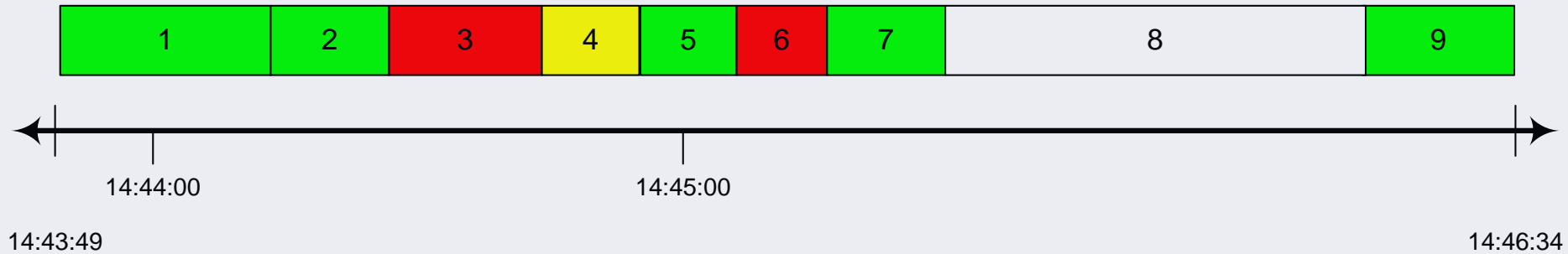
Visualization: Track (Spur)

- Tracks is view/filter/selection of Episodes, i.e. parts of the actual process
- Visualization = showing Tracks
- Selection criteria are Episode's attributes, i.e.
 - Excerpt (i.e. time, people, project)
 - Operation (out of Categories)
 - Reference (to Focus)
 - Link (to other Episodes)
 - Characteristics



Visualization: Example (1)

- First example of a Track:



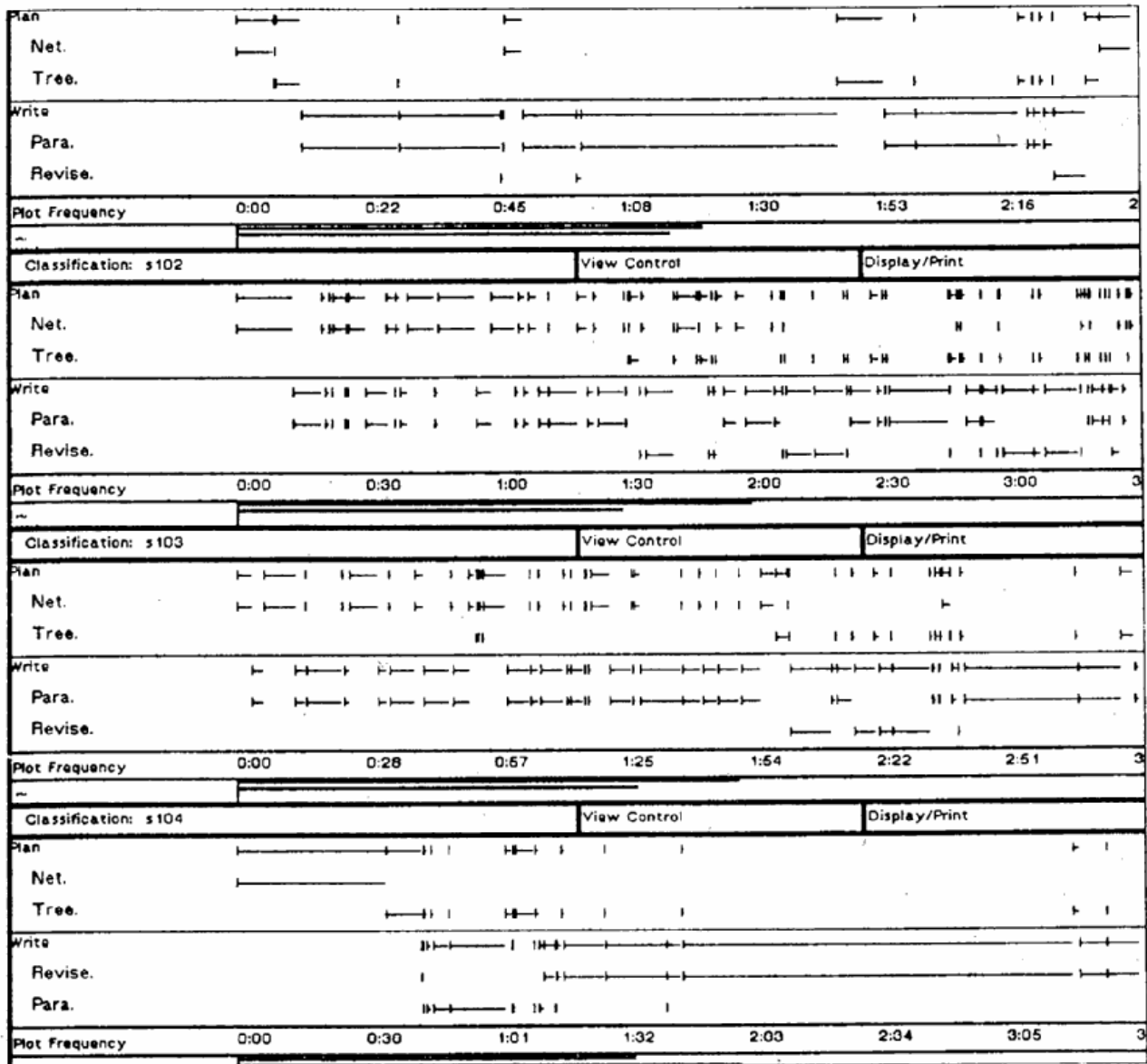
Category Activities

- Legend:

 Advancement	 Browse
 Betterment	 Thinking

- Missing: 1 introduces defect and creates location, 5 leaves location open, 8 includes cursor moving, etc.
- 9 is main example

Figure 7. Abbreviated events-time displays for four subjects, permitting comparison of their different planning and writing strategies.



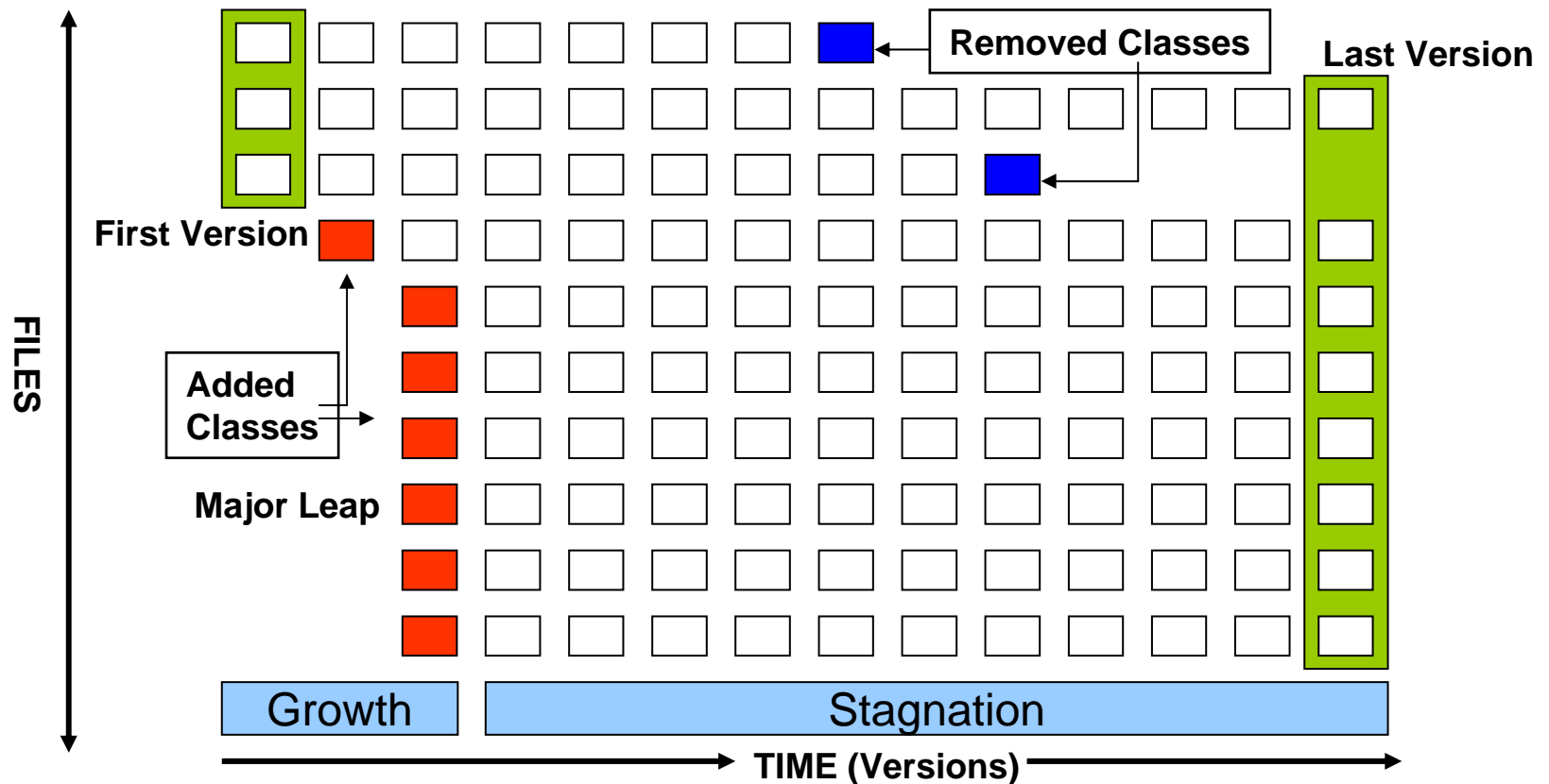
from **Smith, John B.,
Smith, Dana K.,
Kupstas, Eileen
(1993): Automated
Protocol Analysis. In
Human-Computer
Interaction, 8 (2),
pp. 101-145**

Creating a visualization

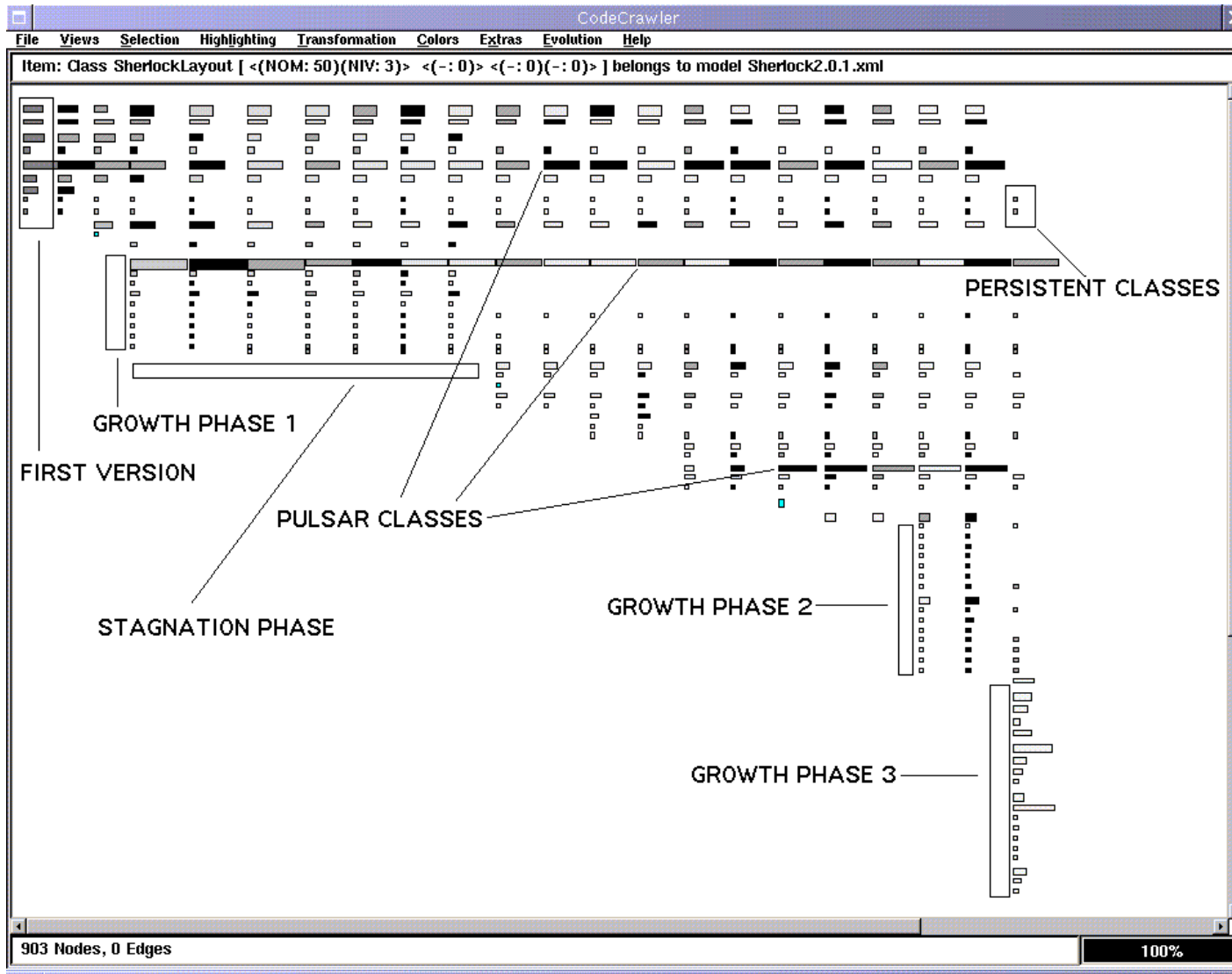
1. Selection = Define Track (i.e. set of Episodes)
2. Projection = Choose attributes to be displayed
3. Decoration = Specify how to display the attributes
 - i.e. type of view
 - Bars (height/width), Text, Colors (Base + Brightness), Texture, Symbols, etc.
 - ... of each dimension
 - Characteristics, Activity, Excerpt length, etc.
4. Repetition = Different Tracks can be viewed one below the other
 - for similar time frame
 - especially different projections for the same selection

Related work: Evolution Matrix (1)

- What about Foci? Idea: A Track per Focus.
- Similar design: Evolution Matrix

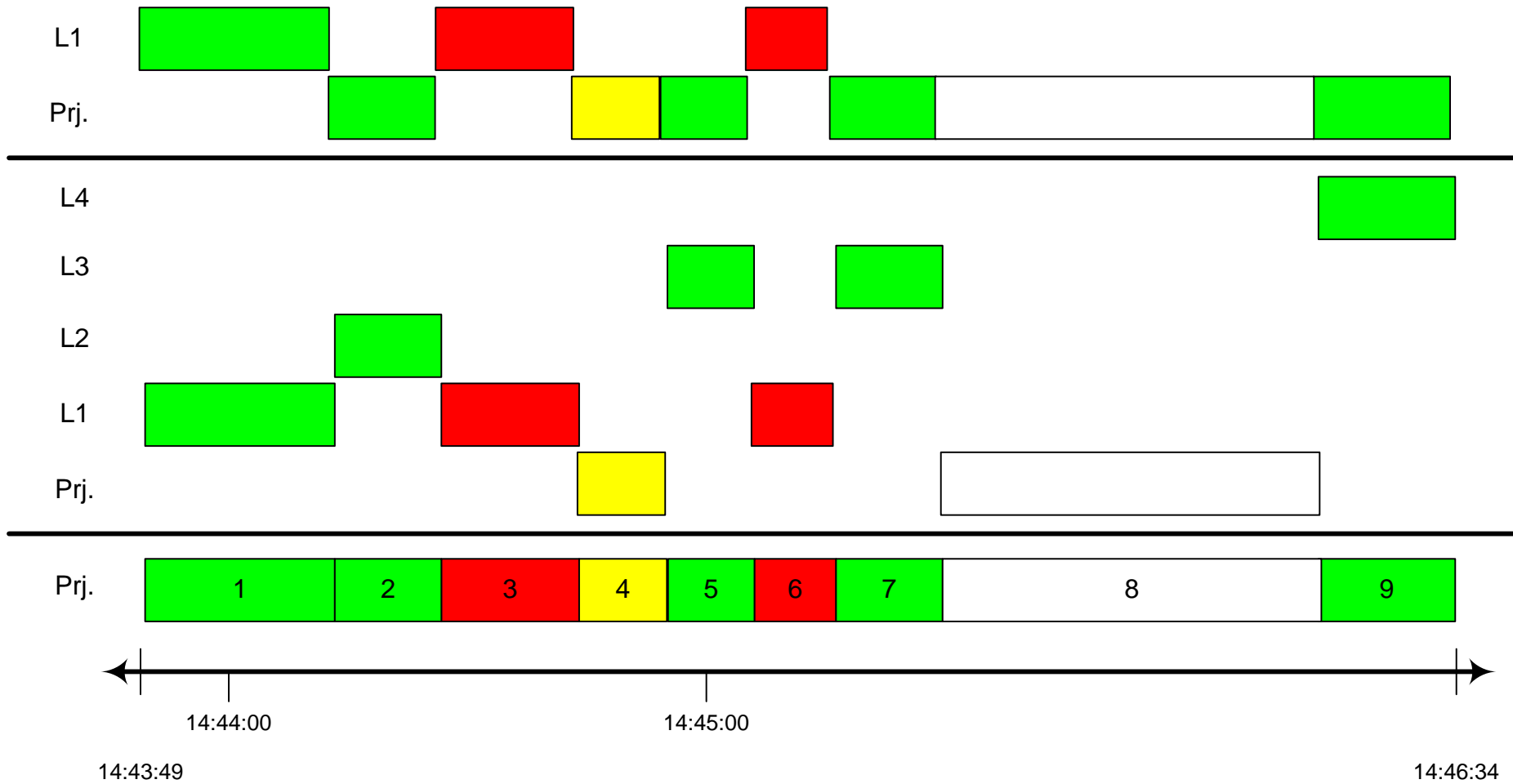


Related work: Evolution Matrix (2)



from:
Michele Lanza:
The Evolution Matrix:
Recovering Software
Evolution using
Software Visualization
Techniques, 2001

Visualization Example (2)



Thank you!