Recurrent Events: Comparing and Explaining Behaviour

Abram Hindle, Micheal W. Godfrey, Richard C. Holt

Software Architecture Group
David R. Cheriton School of Computer Science
University of Waterloo
Canada
http://swag.uwaterloo.ca/

{ahindle,migod,holt}@cs.uwaterloo.ca

Introduction

- Care about recurrent events
- Events which repeat
- Care about events which induce behaviours
  - or are part of a behaviour
    - Can we find the reason for a behaviour

Self Similarity

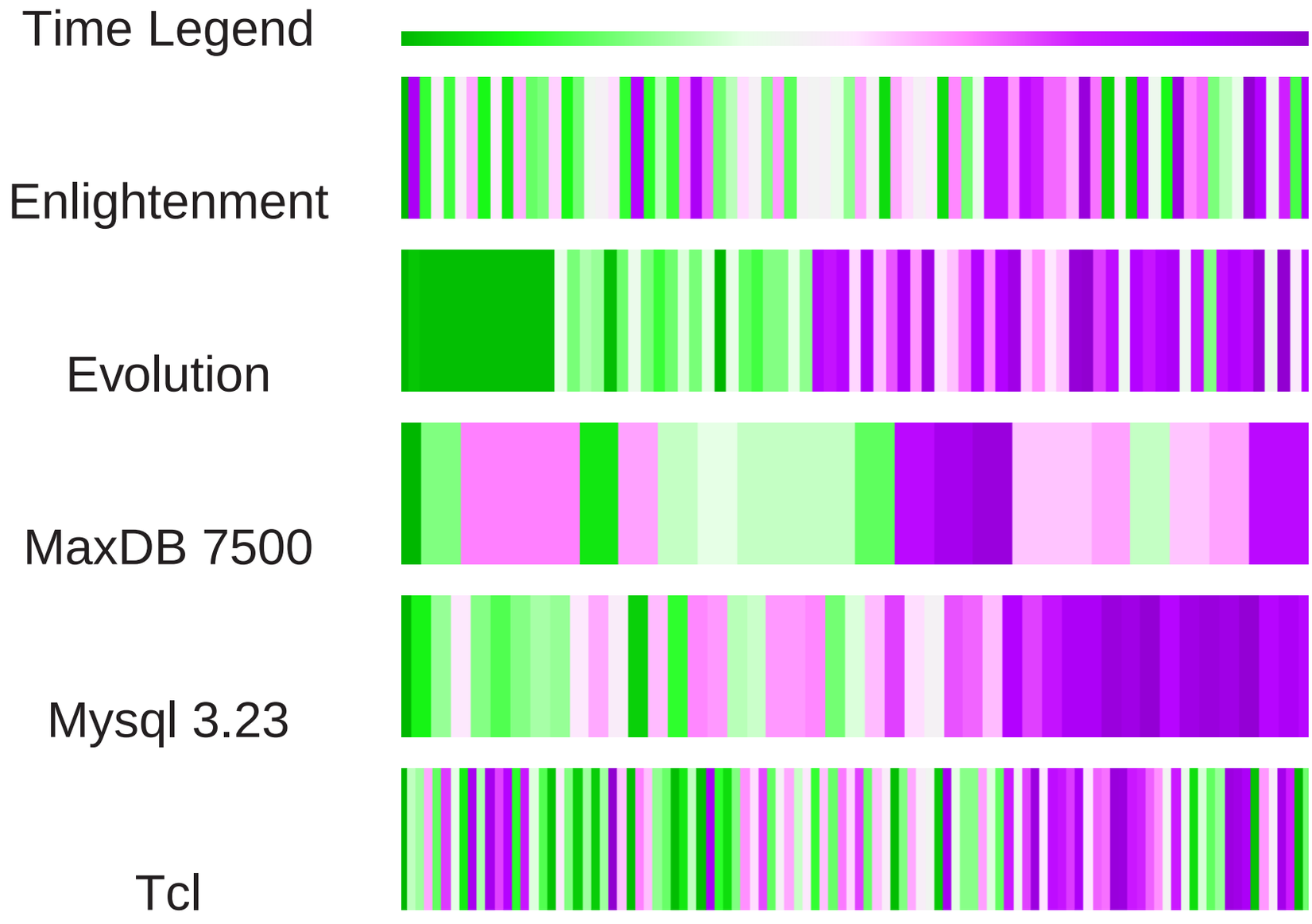
- Take regions of a project's lifetime, compare their similarity to other regions
  - Look for continuous regions that are relatively similar
  - When behaviour changes we can partition at that point
    - Different regions of different behaviour could indicate:
      - An iteration
      - A milestone
      - A sub-part of an iteration
      - Freezes etc.

Self Similarity Methodology

- Extract a repository
- Break down into counts by unit
- Choose a Fourier Transform size (window)
- Per each window apply the Fourier transform
- Per each window determine all the similar windows
- Look for continuous regions which repeat ..

Self Similarity Plot

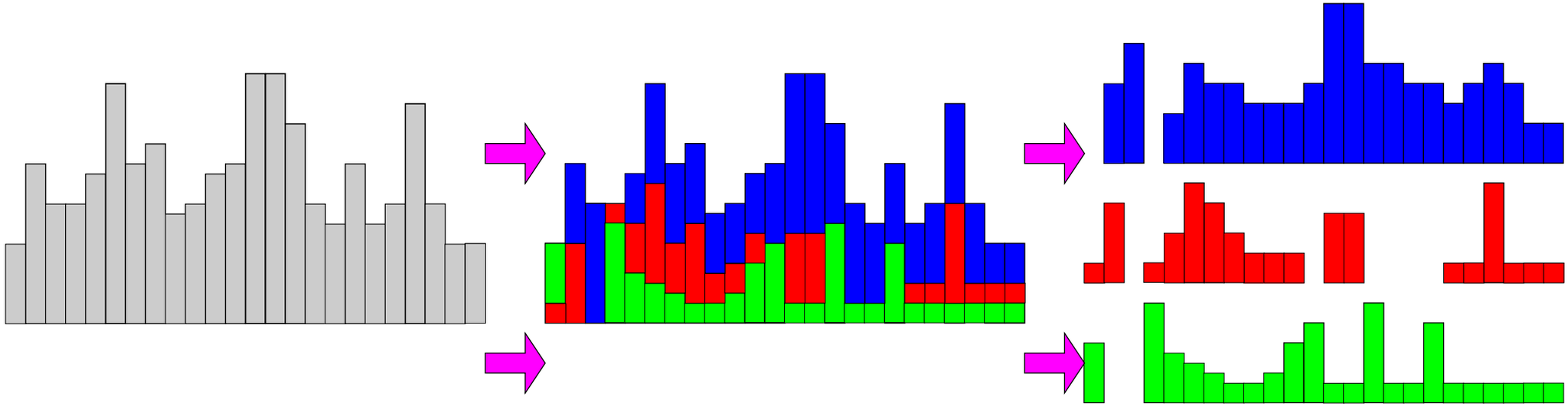
- Extract and compare windows
- Choose a color palette
- By default each window has its own color from the palette
- If a window is similar to another window color color it that color
  - Choose an order like for a set of similar windows they receive the color of the earliest window



Self Similarity of multiple OSS projects (similarity to other month-sized window shown by color)

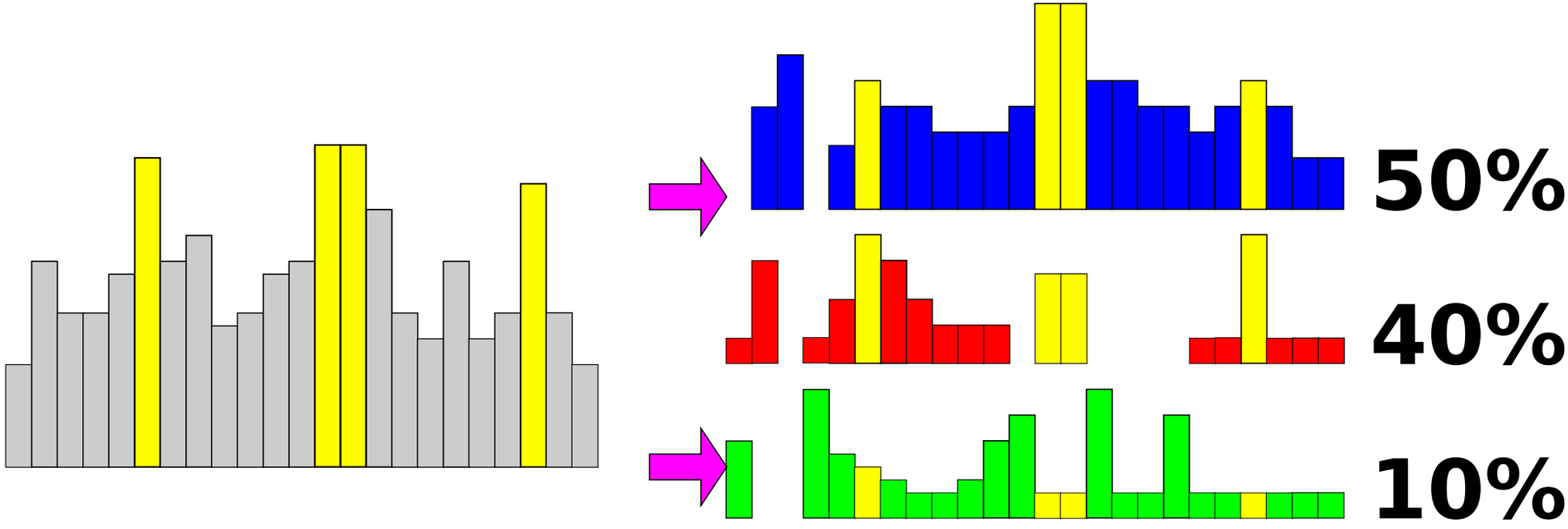
Slicers

- Idea: We can split up streams into slices which we can describe
- A slice we can easily describe
  - Authors
  - Files
  - Modules
  - ...
- If behaviour is induced by such a slice we can easily describe which data was at fault
  - “Look this spike with a frequency of 30 days is caused by the European localization team”



A signal composed of slices

A signal can be composed of slices



A behaviour explained by slices

Different slices can be responsible for sub-signals

Case Study: MaxDB 7.500

- Using Directory based slices
  - changes dir ranked very high in “individual best”
  - changes ranked last in leave one out analysis
    - This means that files or modules which frequent changes might have reasonable distance between said behavior
    - But they might not actually take part
      - Their magnitude can give the illusion of participating
- The most commonly modified directory ranked high by individual distance to top 3 peak behaviour