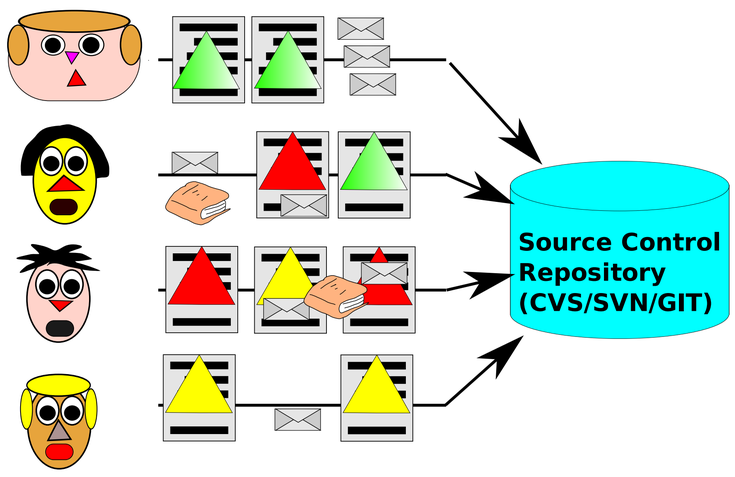
Mining Recurrent Activities: Fourier Analysis of Change Events

Abram Hindle, Michael Godfrey, Ric Holt
University Of Waterloo, Waterloo, Canada

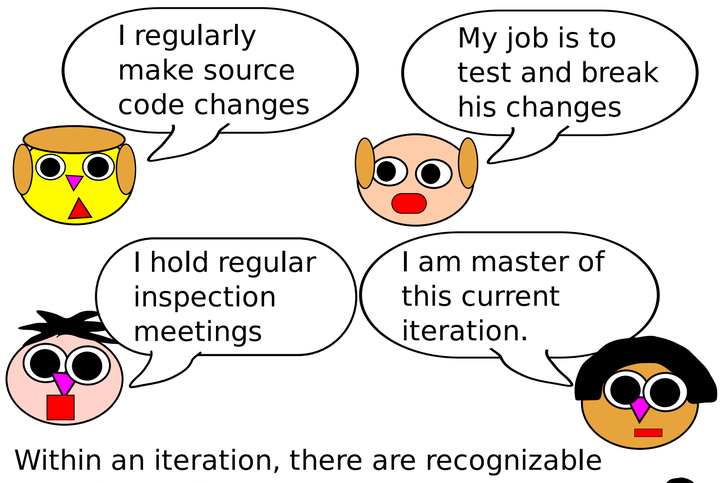
Developers create artifacts



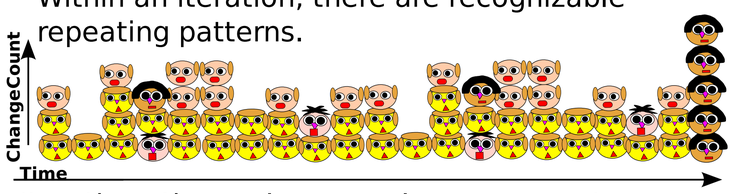
Developers create various kinds of artifacts when they make changes:

source code, test suites, bug reports, documentation, mailing list messages, etc

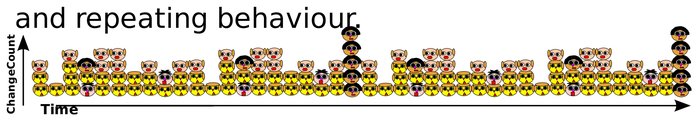
Software development has recurrent behaviour



Within an iteration, there are recognizable repeating patterns.

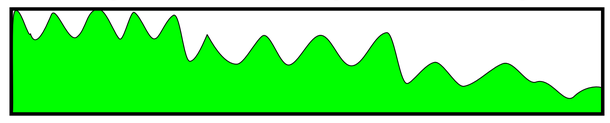


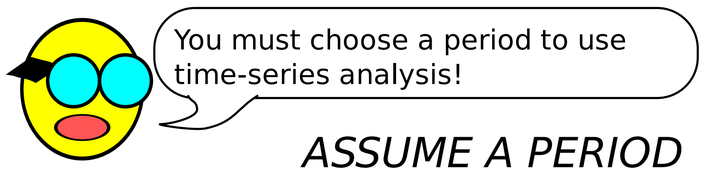
Iterations themselves are also a recurrent and repeating behaviour.



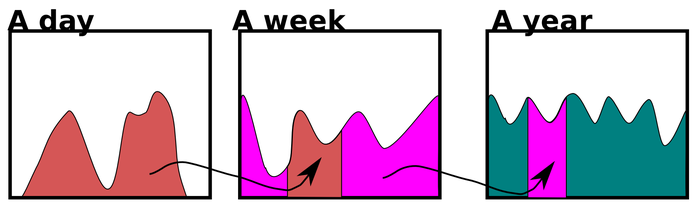
How do we discover recurrent behaviour? With Time-series analysis

A signal, multiple years in length

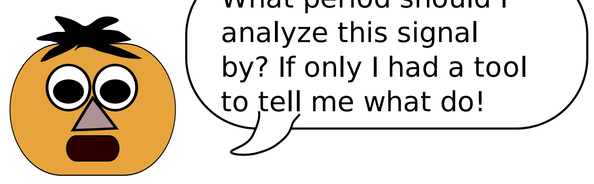




ASSUME A PERIOD

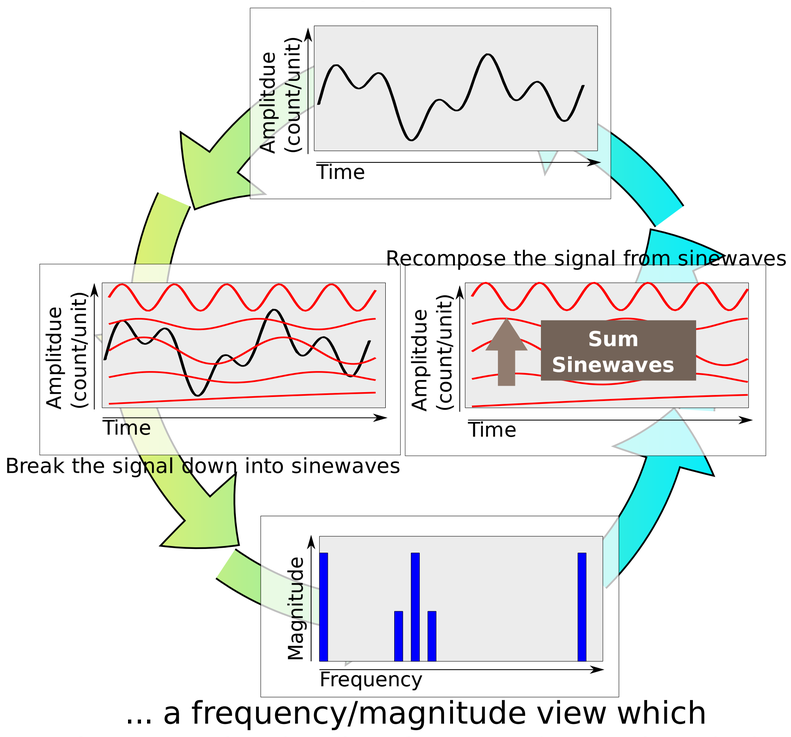


But which is correct or useful?



A solution: Fourier transform

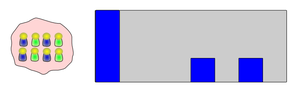
We can convert from a time/amplitude view to...



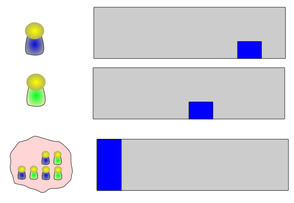
... a frequency/magnitude view which shows us dominating frequencies and periods

How can we apply the Fourier transform to software change and software related data?

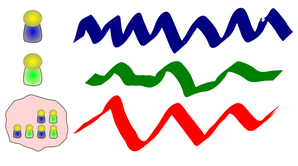
We have a signal



We apply the Fourier transform.

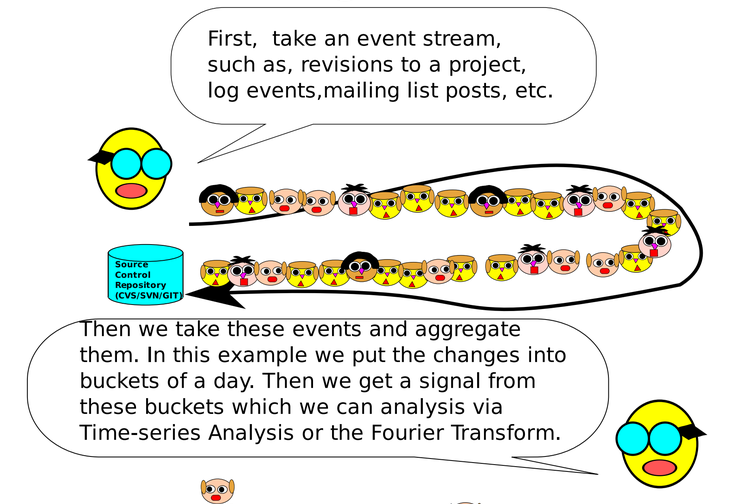


Investigation of each frequency bin shows that 2 authors are individually responsible for 2 bins. Their frequencies imply their periodicities.

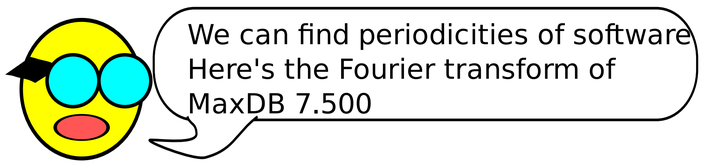


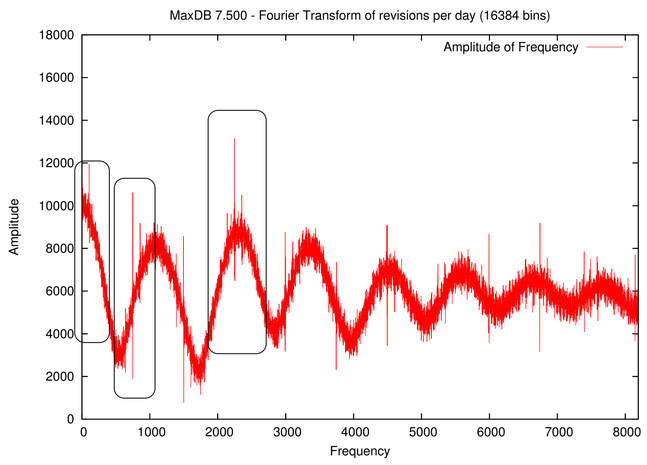
We discover that:
* blue commits many changes
* green tests blue's commits
* The other developers follow their own schedule.

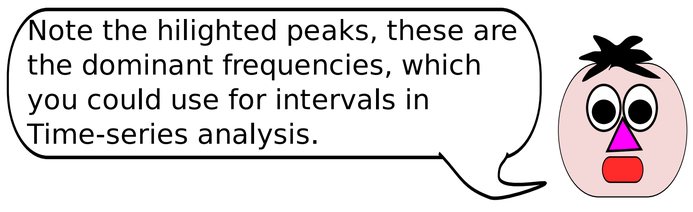
Convert discrete events into signals



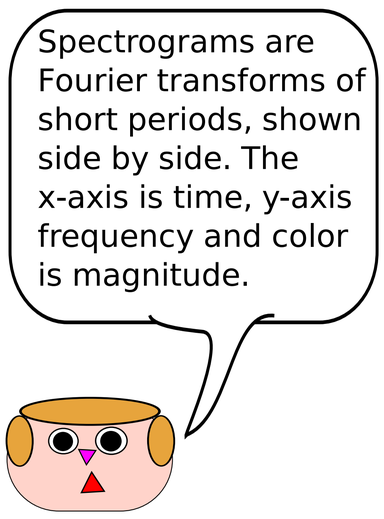
So what? Does it work?

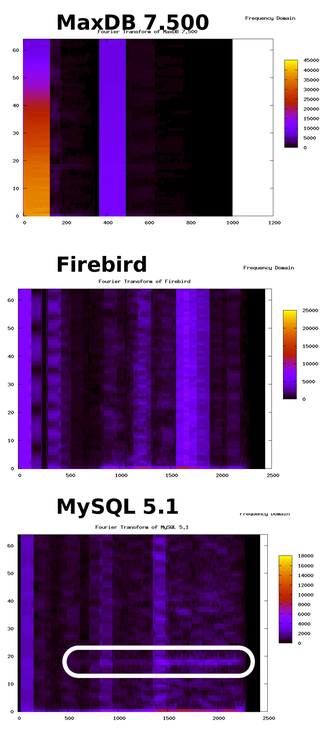






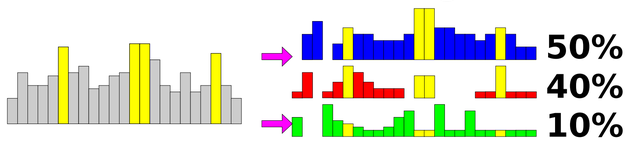
Spectrograms of software change over time





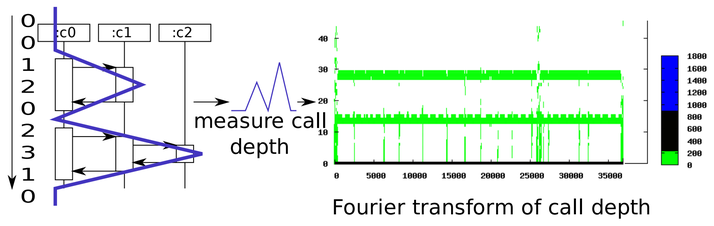
Future Work

Who's responsible for this behaviour? Semantic Slicing



A behaviour can be explained by a combination of slices.

Other uses of the Fourier Transform: Dynamic Analysis via call depth



Partition development time via Self Similarity