



**HAL**  
open science

## Analysis and Advanced Visualization of Software

Pierre Caserta, Olivier Zendra, Damien Bodénès

► **To cite this version:**

Pierre Caserta, Olivier Zendra, Damien Bodénès. Analysis and Advanced Visualization of Software. The 25th European Conference on Object-Oriented Programming (ECOOP 2011), Jul 2011, Lancaster, United Kingdom. inria-00612601v2

**HAL Id: inria-00612601**

**<https://inria.hal.science/inria-00612601v2>**

Submitted on 21 Aug 2011

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Analysis and Advanced Visualization of Software

## The Problem:

Software is complex and contains large amounts of information. This makes understanding its design, functionality and behavior a difficult and time-consuming task.

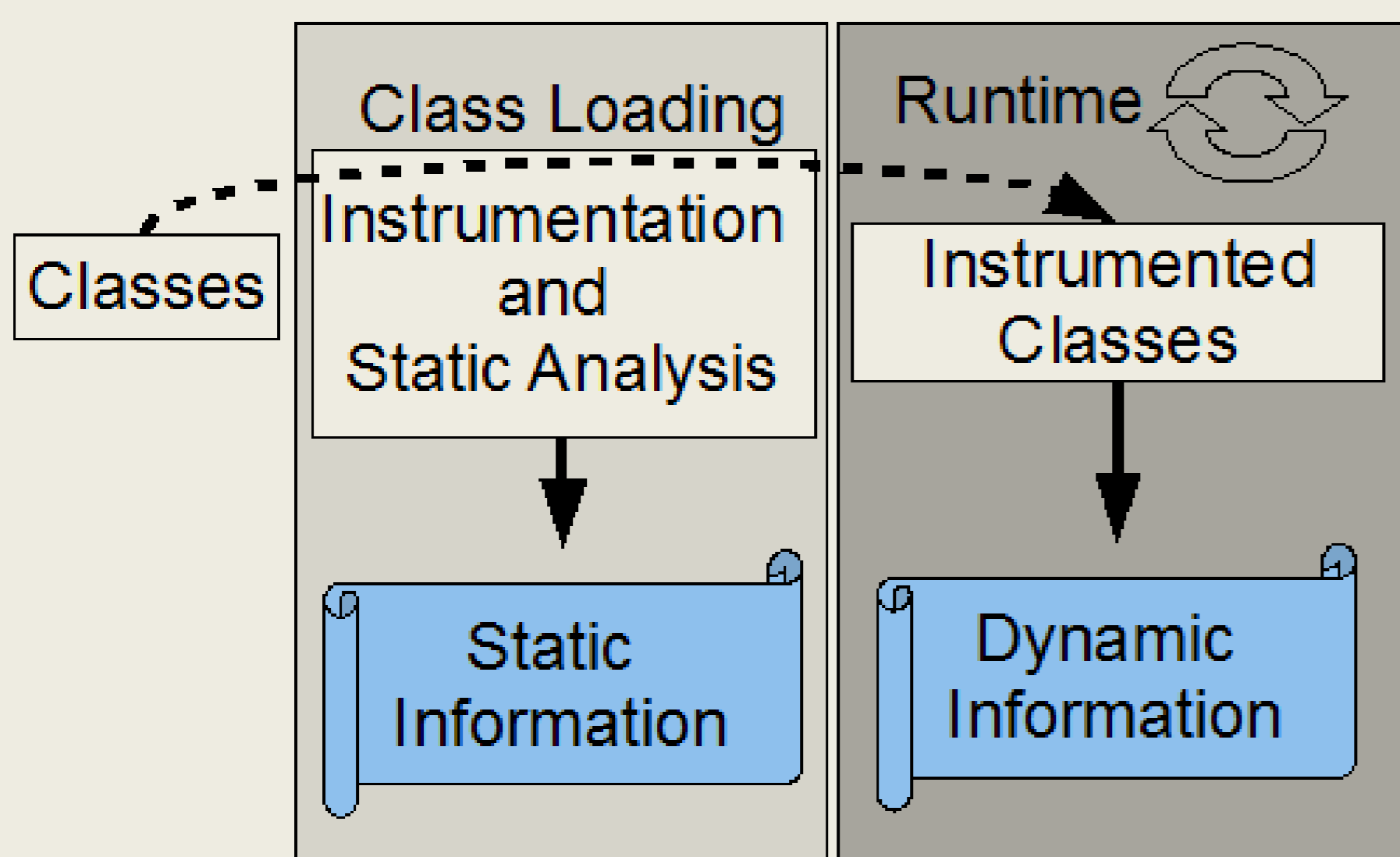
## A Solution:

Use analysis techniques to extract and compute useful metrics on the software system. Then represent these metrics through suitable abstractions and metaphors to ease the understanding of the software.

## Tracing Software with VITRAIL

### JBInsTrace:

Instrument bytecode to collect static as well as dynamic information on the software.



## Analyzing Software with VITRAIL Analyzer:

### Static Information:

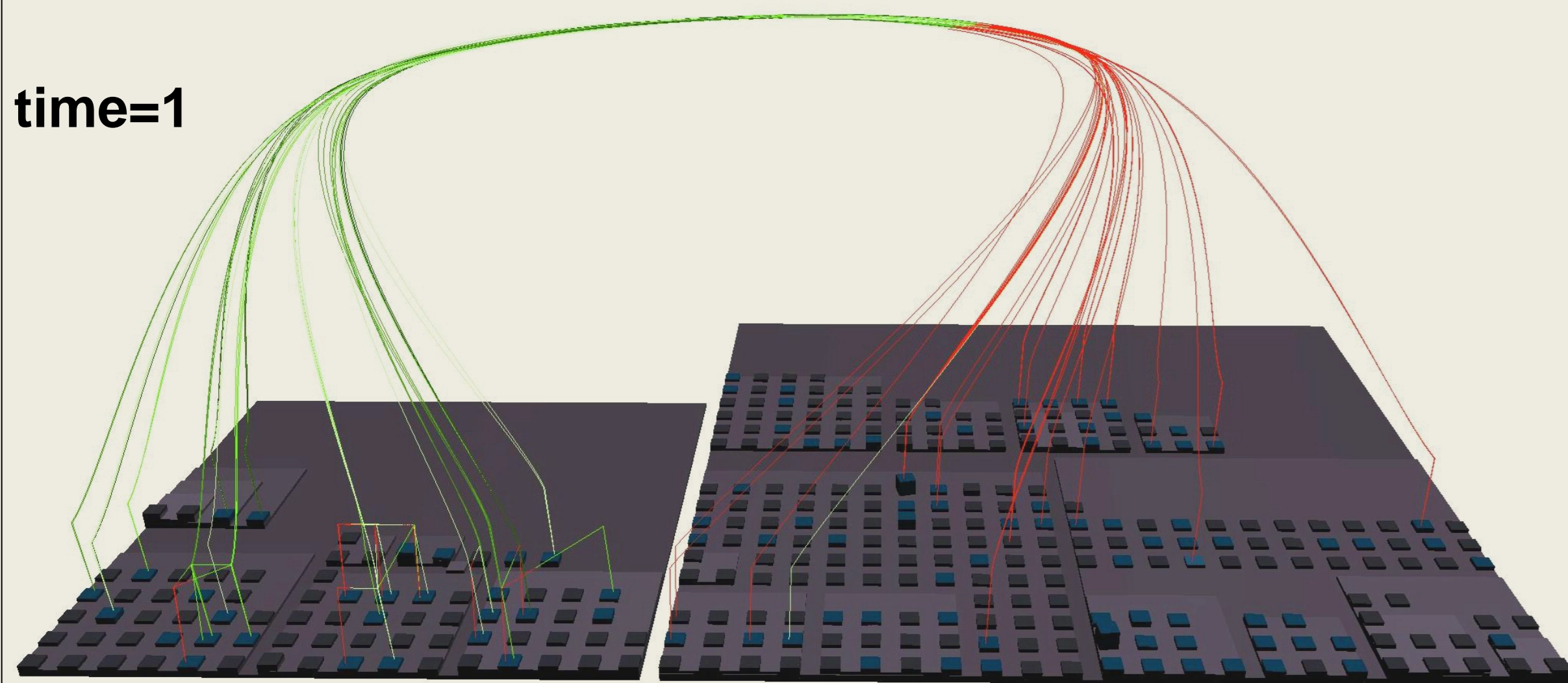
- Software architecture (inheritance, packages, relations ...).
- Static receiver type on call-sites.
- Static metrics (number of bytecodes, number of if-statements ...).

### Dynamic Information:

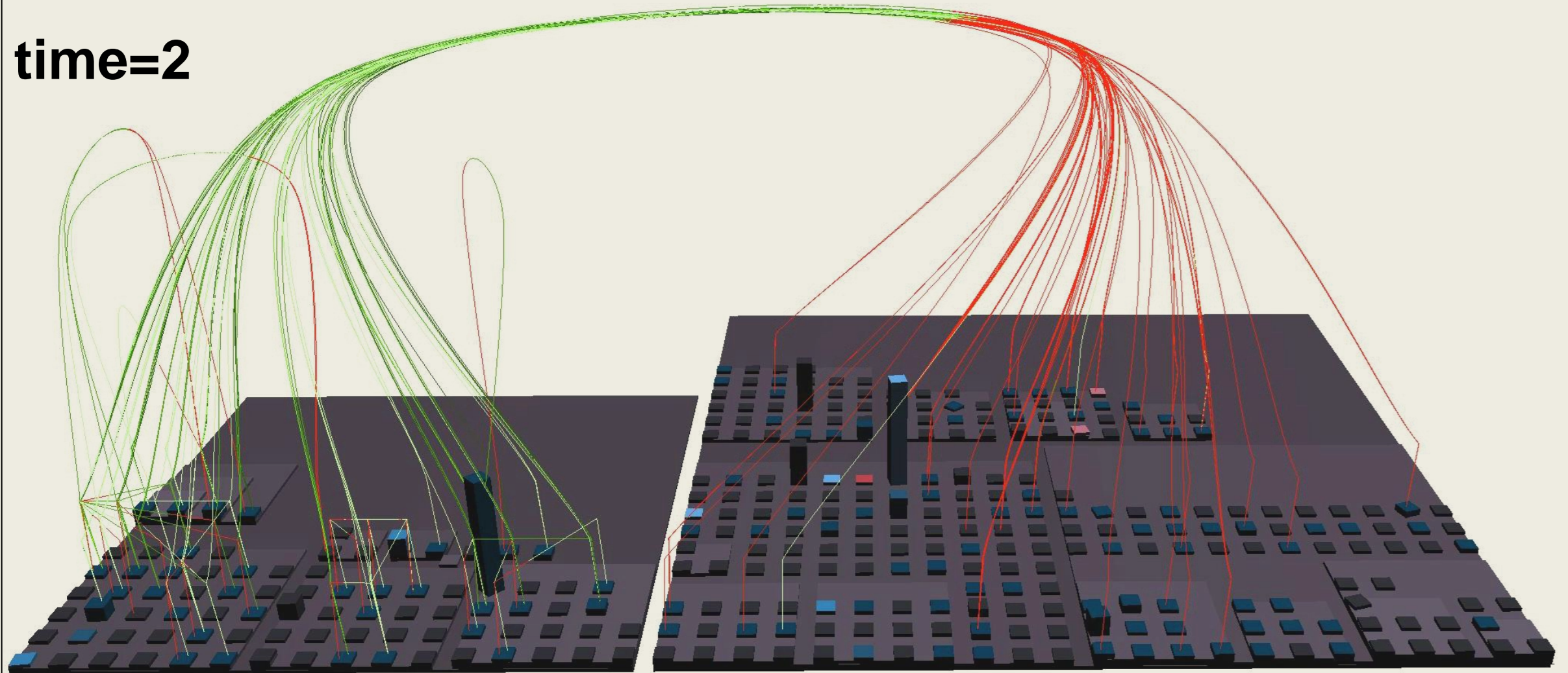
- Control flow (at basic block level).
- Dynamic instance type used on call-sites.
- Dynamic metrics (number of executed bytecodes, number of if-statements taken ...).

## Visualizing Software Evolution with VITRAIL Visualizer:

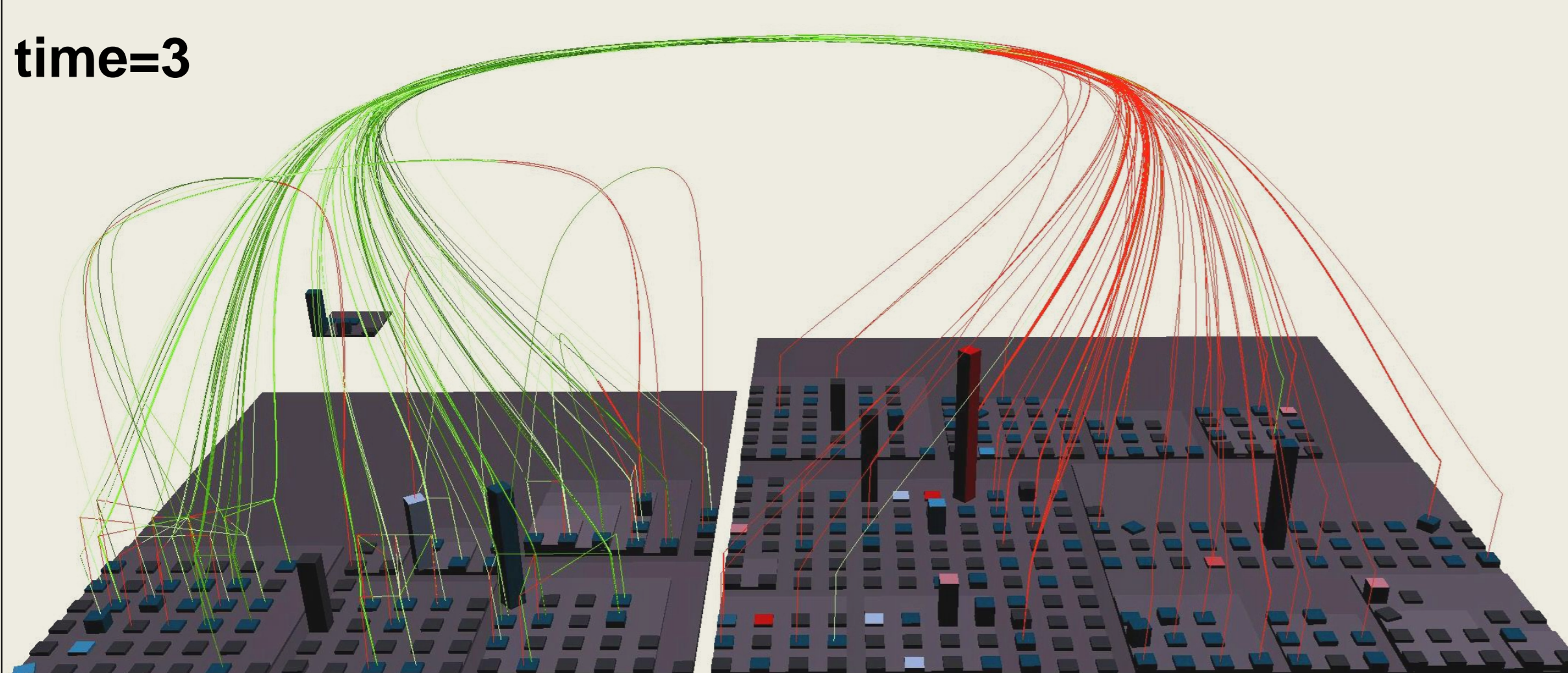
time=1



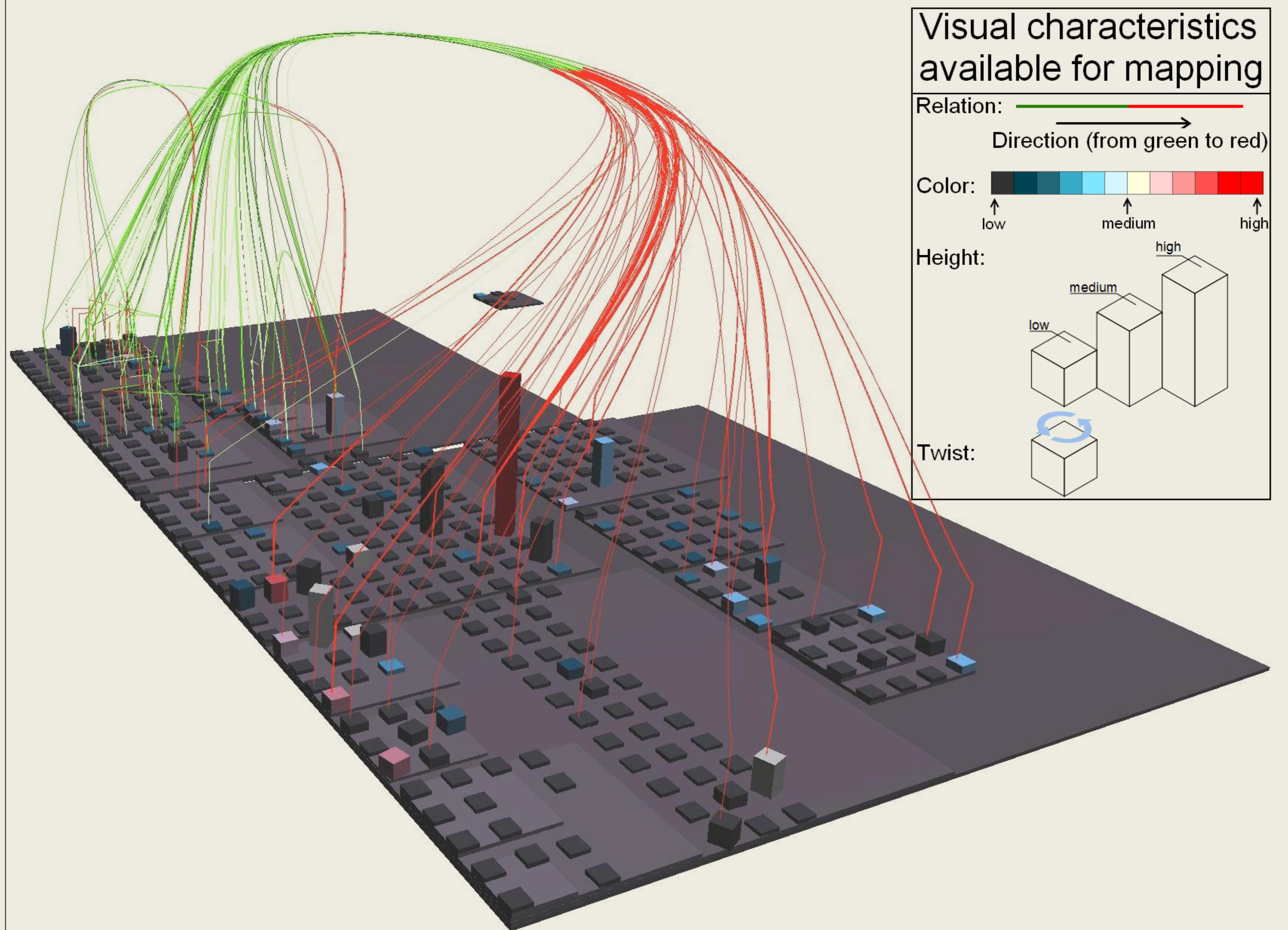
time=2



time=3



## Visualizing Software with VITRAIL Visualizer:



## Understanding Software Functioning:

- Static relations between software components.
- Dynamic relations between software components.
- Evolution of these dynamic relations during runtime.
- Impact of each component on the whole system.

## Purpose :

- Design and Source code optimizations.
- Improve maintenance process.
- Study the habits of object-oriented developers.

## Current Work:

- We perform tracing and analysis on many pieces of software to get useful static and dynamic information.
- We study how software was designed and its behavior at runtime using VITRAIL Visualizer.
- We use new interaction and display technologies to ease manipulation and understanding of the visualization.