The Deconstruction of Dyninst: Experiences and Future Directions

Drew Bernat, Madhavi Krishnan,
Bill Williams, Bart Miller
Paradyn Project

Why components?

Share tools

Build new tools quickly

Share Tools

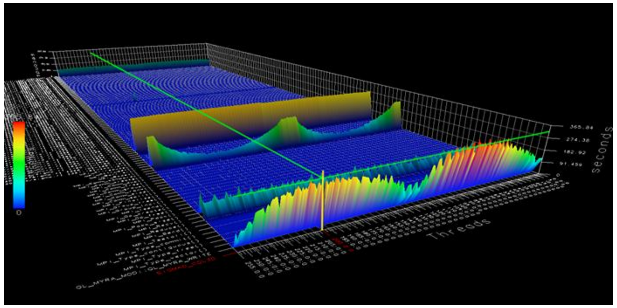
scalasca

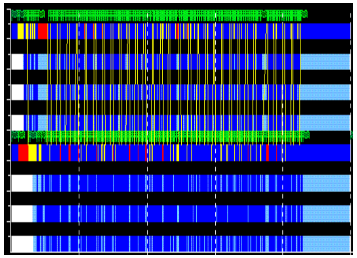
τ

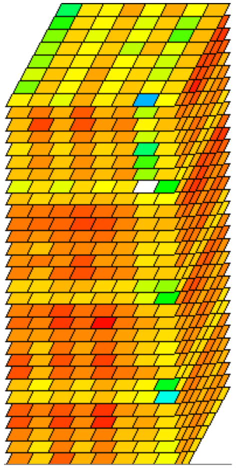
Dyninst

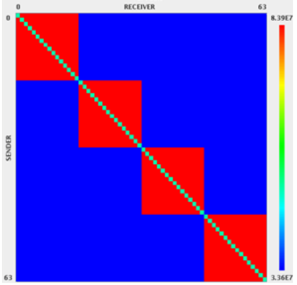
Open | SpeedShop™

Paraver









Dyninst Components

DyninstAPI

Dyninst Component Users

StackwalkerAPI
A Dyninst Component

SymtabAPI
A Dyninst Component

Stack
Trace
Analysis
Tool

2011 R&D 100

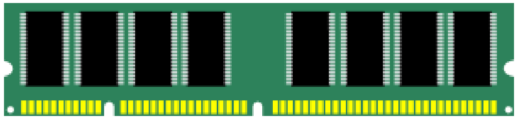
CRAY ATP

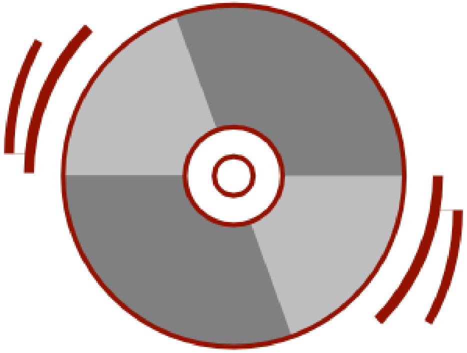
HPCToolkit
Rice University

Build New Tools Quickly: Dataflow Analysis

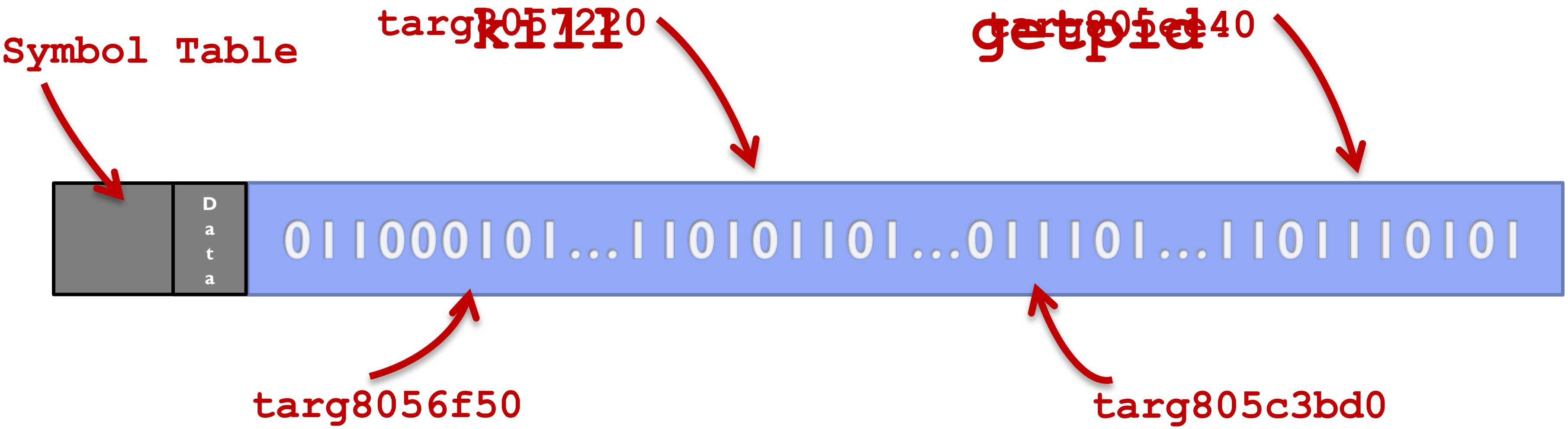
- PowerPC jump tables and return instruction detection
- Malware return address tampering
- Behavior-preserving relocation

Build New Tools Quickly: Binary Rewriter





Build New Tools Quickly: Unstrip



Symbol Table

D a t a

011000101...110101101...011101...1101110101

targ8057220

kill

targ805ed40

getpid

targ8056f50

targ805c3bd0

ParseAPI

A Dyninst Component

ROSE

compiler

SymtabAPI

A Dyninst Component

Down The Memory Lane

SymtabAPI - version 1.0

DynStackwalker - coming soon

InstructionAPI - proposed

BinInst - proposed

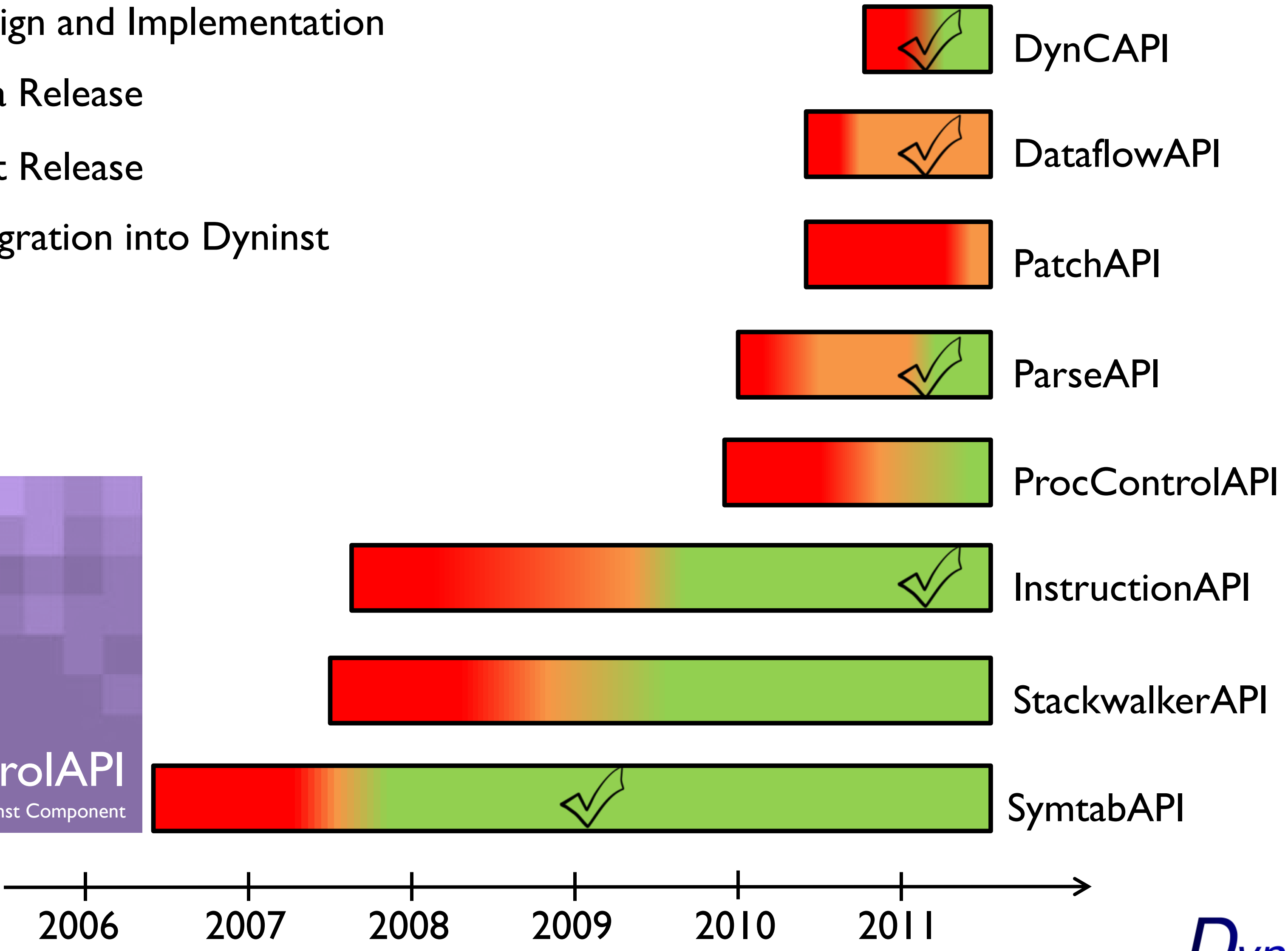
Dyninst Components Timeline

🟥 Design and Implementation

🟧 Beta Release

🟩 First Release

✓ Integration into Dyninst



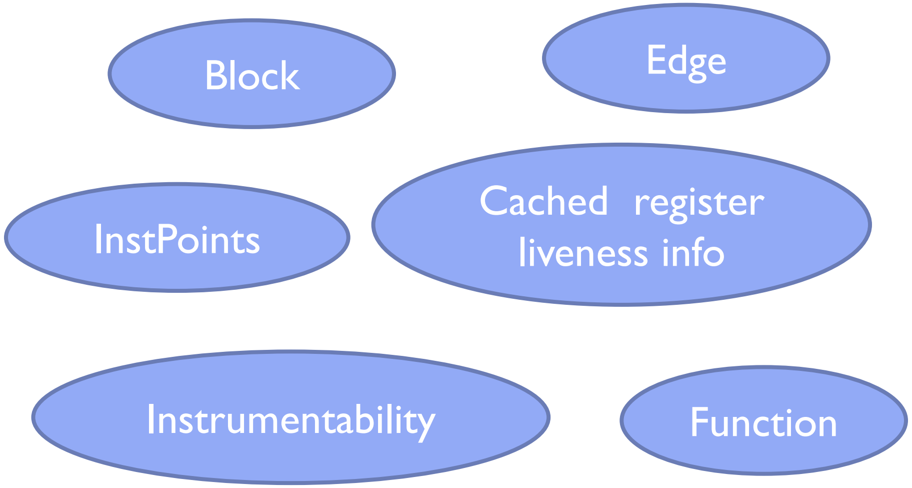
ProcControlAPI
A Dyninst Component

Componentization: Design Decisions

Define the scope of the component

ParseAPI CFG model

Dyninst CFG model



Block

Edge

InstPoints

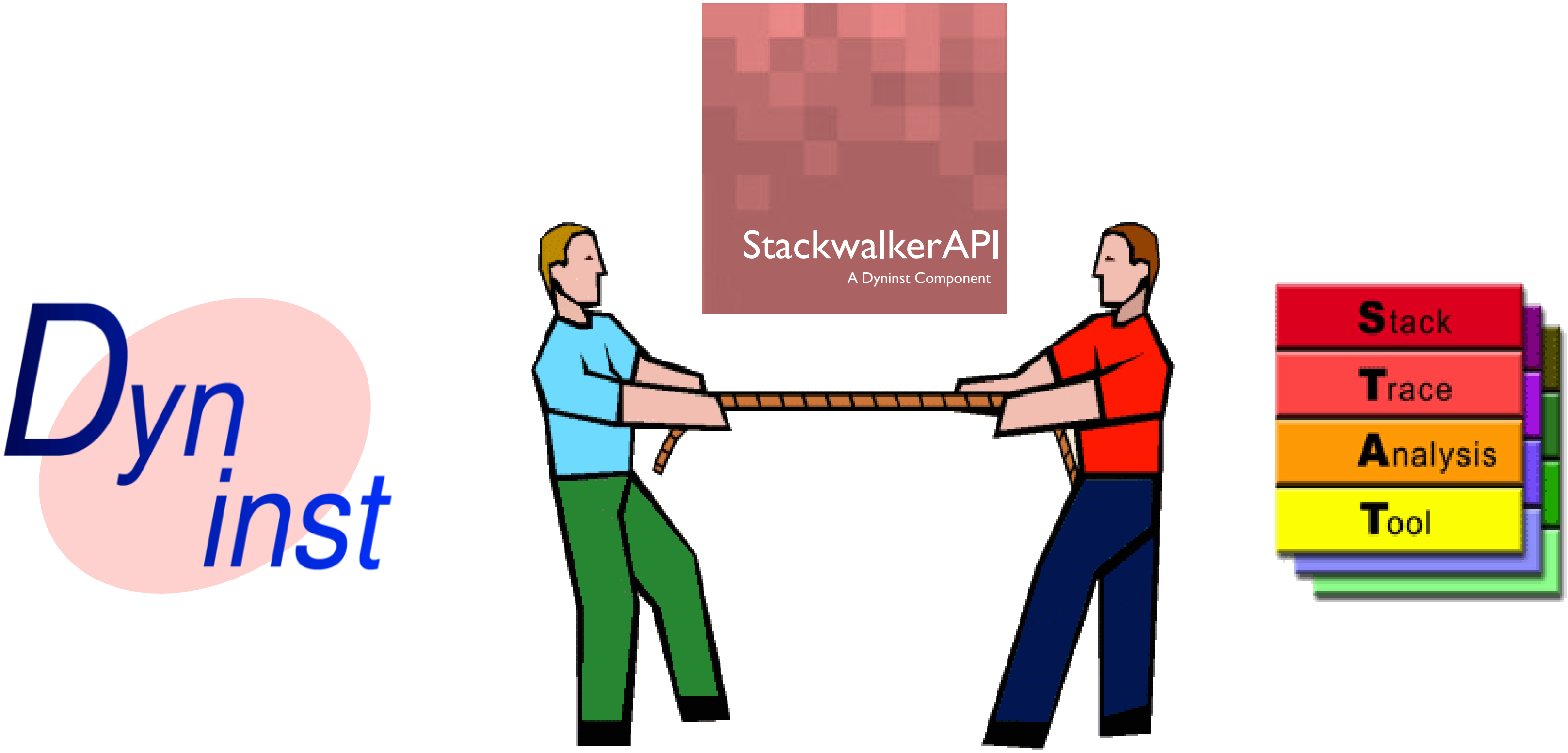
Cached register liveness info

Instrumentability

Function

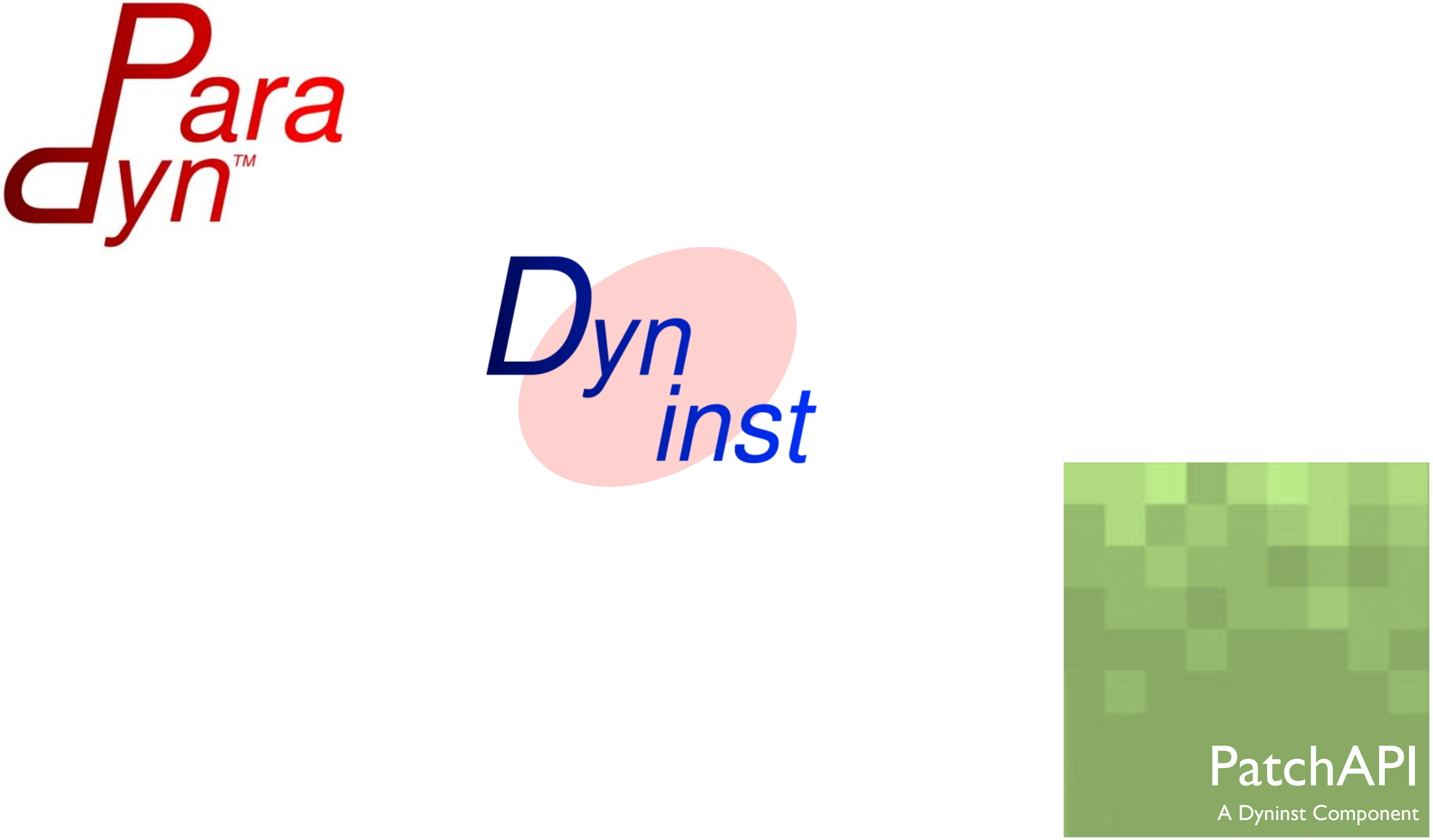
Componentization: Design Decisions

Balance internal and external user requirement



Componentization: Design Decisions

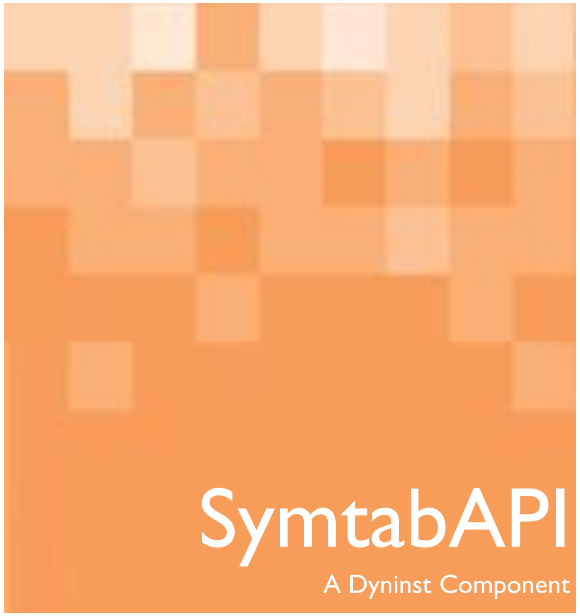
Refine requirements



Componentization: Design Decisions

Create right level of abstractions

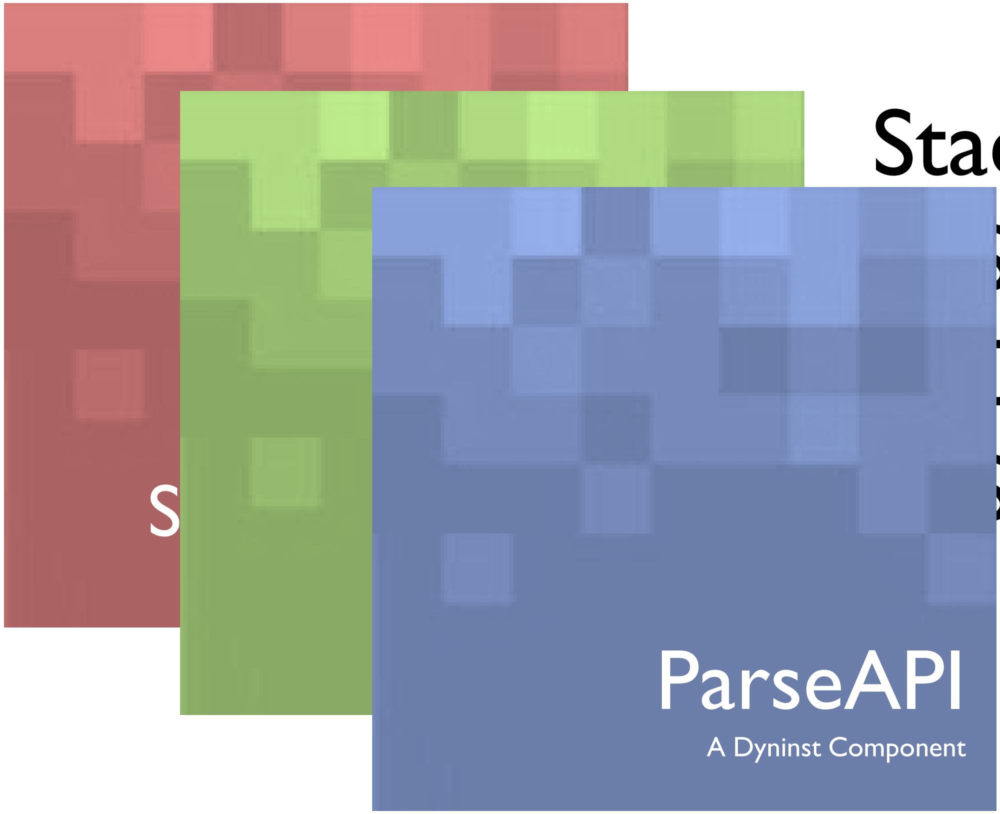
libbfd



libelf

Componentization: Design Decisions

Design extensible and adaptable interfaces



Stack frame stepper

- Standard frame
- Debug frame
- Signal frame

Componentization: Design Decisions

Plan for reintegration

Ongoing Research

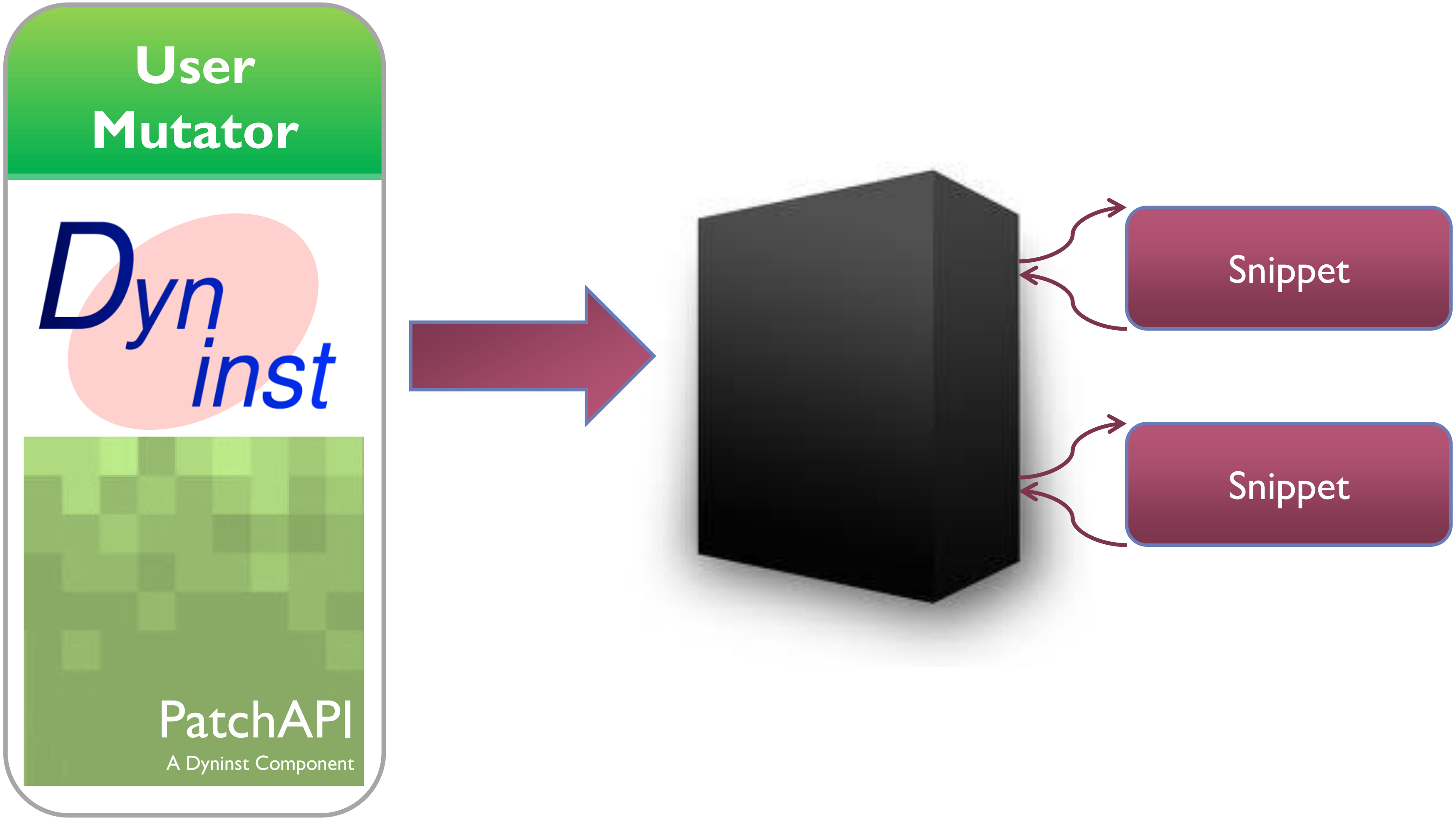
Ongoing Research

- Lightweight, Self-Propelled Instrumentation
  - Wenbin Fang
- Binary Editing
  - Andrew Bernat
- Malware Analysis and Instrumentation
  - Kevin Roundy
- Binary Provenance and Authorship
  - Nate Rosenblum
- Instrumenting Virtualized Environments
  - Emily Jacobson

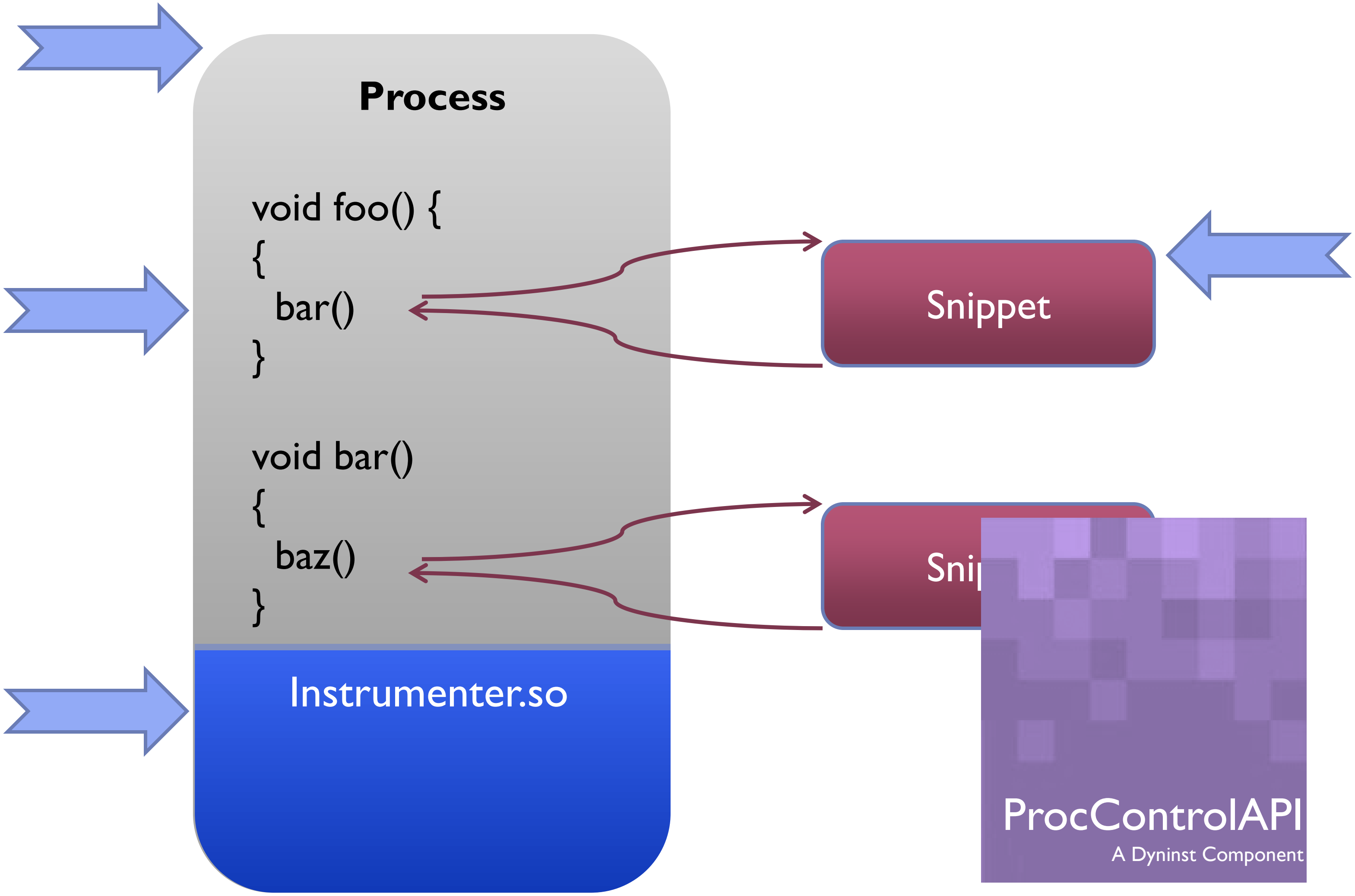
Lightweight Instrumentation

- Analyze intermittent bugs and fine-grained performance problems
  - Autonomy
  - Little perturbation
  - High level of detail
- Rapid activation
- Ability to analyze black-box systems
  - User level and kernel level

Self-Propelled Instrumentation



How it Works



Process

```
void foo() {
{
  bar()
}

void bar()
{
  baz()
}
```

Instrumenter.so

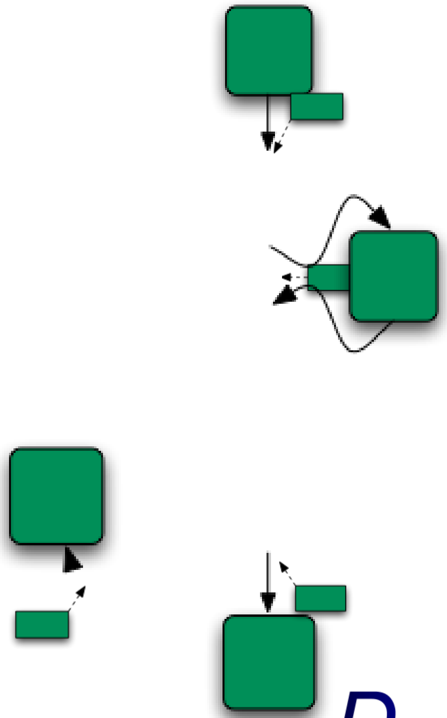
Binary Instrumentation

ParseAPI

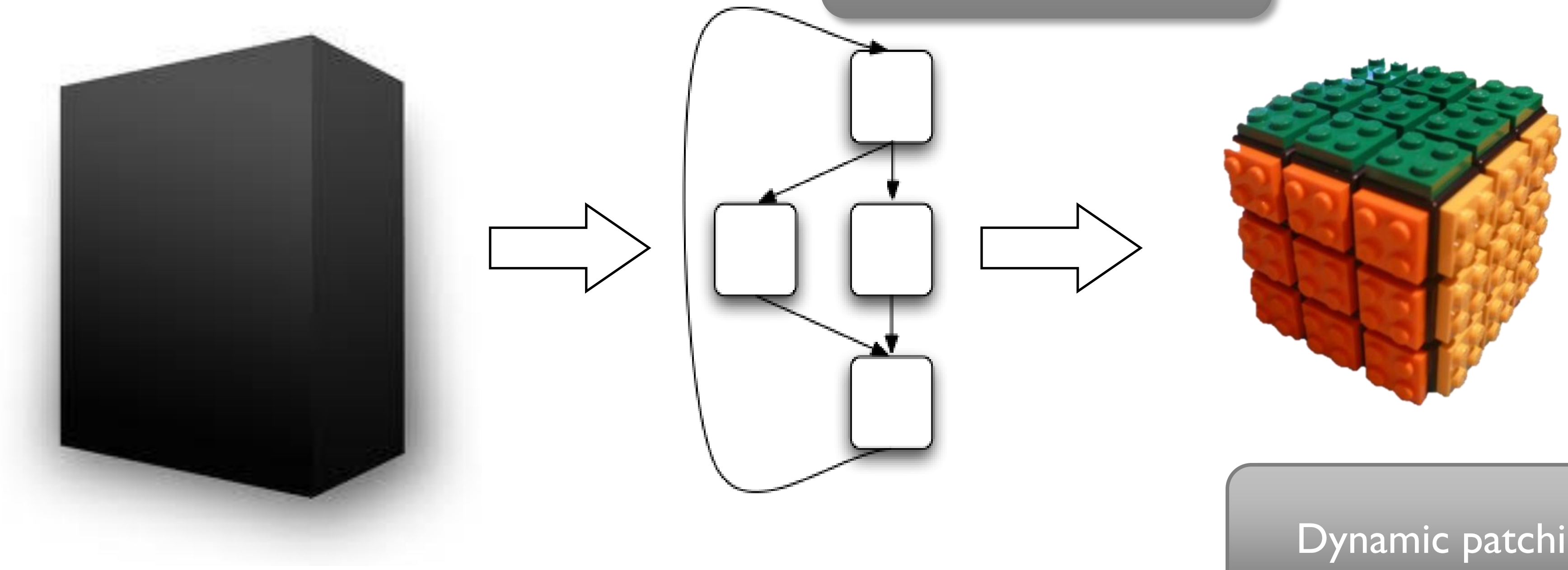
A Dyninst Component

PatchAPI

A Dyninst Component



Binary Editing



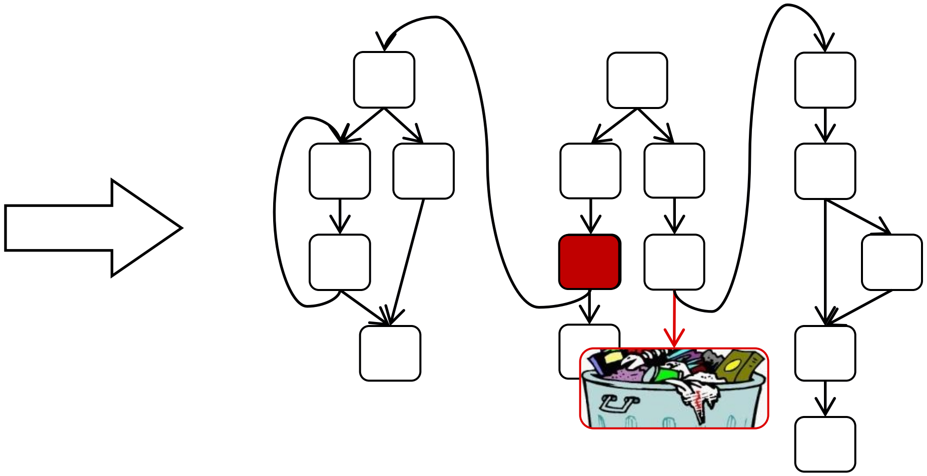
Predicate switching

Insert error checking and handling

Dynamic patching

Code surgery

Malware Analysis and Instrumentation



Unpacking Code

Address Space Sensitive

Overwriting Code

Self-Checksumming

SR-Dyninst

Parse Reachable Code

Dynamic Code Discovery

Catch Exceptions

Overcome Sensitivity

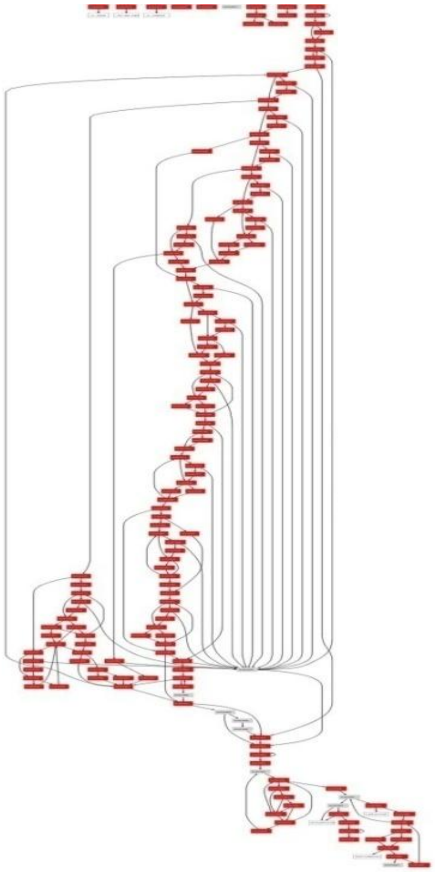
ParseAPI
A Dyninst Component

PatchAPI
A Dyninst Component

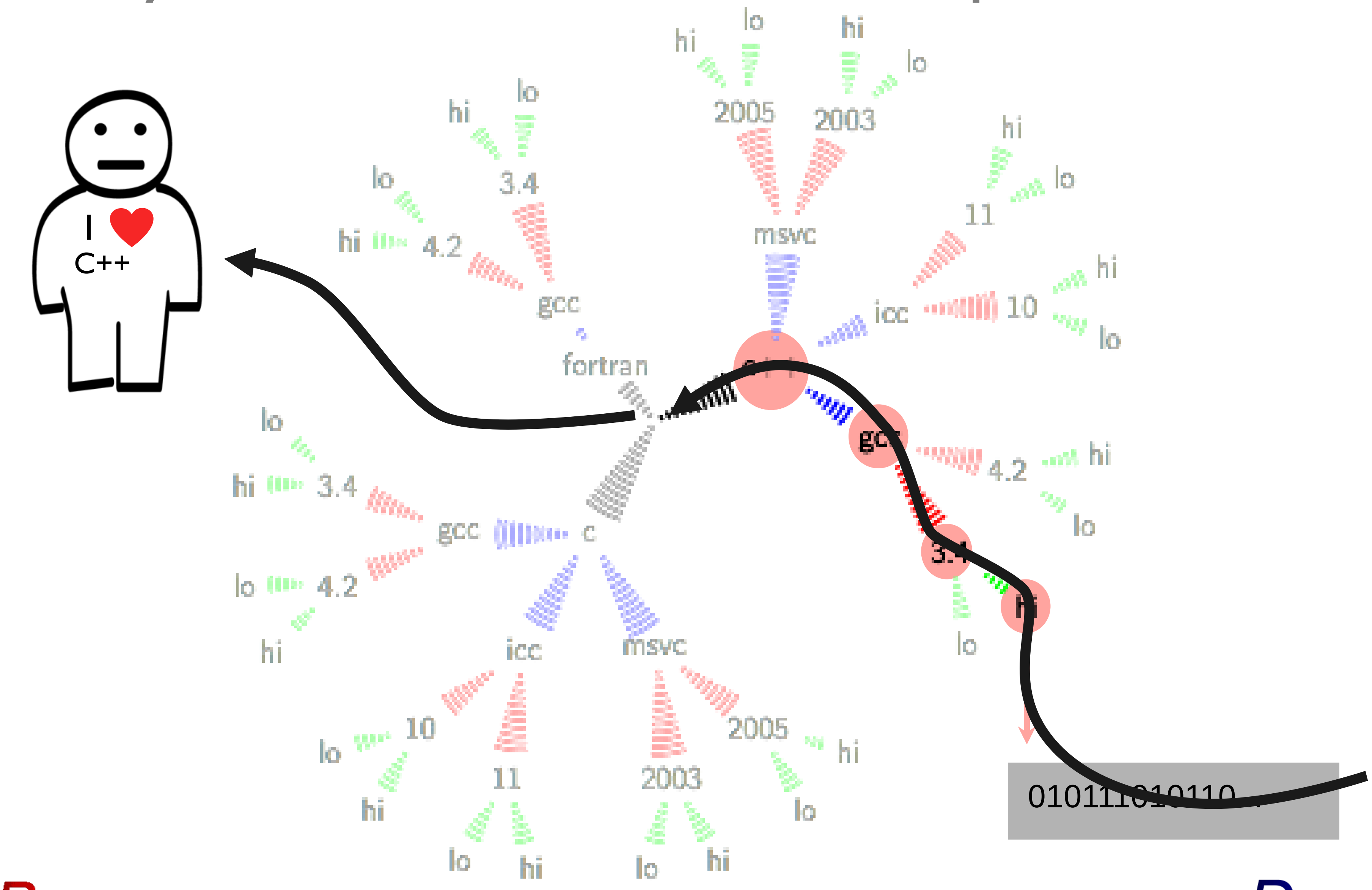
ProcControlAPI
A Dyninst Component

DataflowAPI
A Dyninst Component

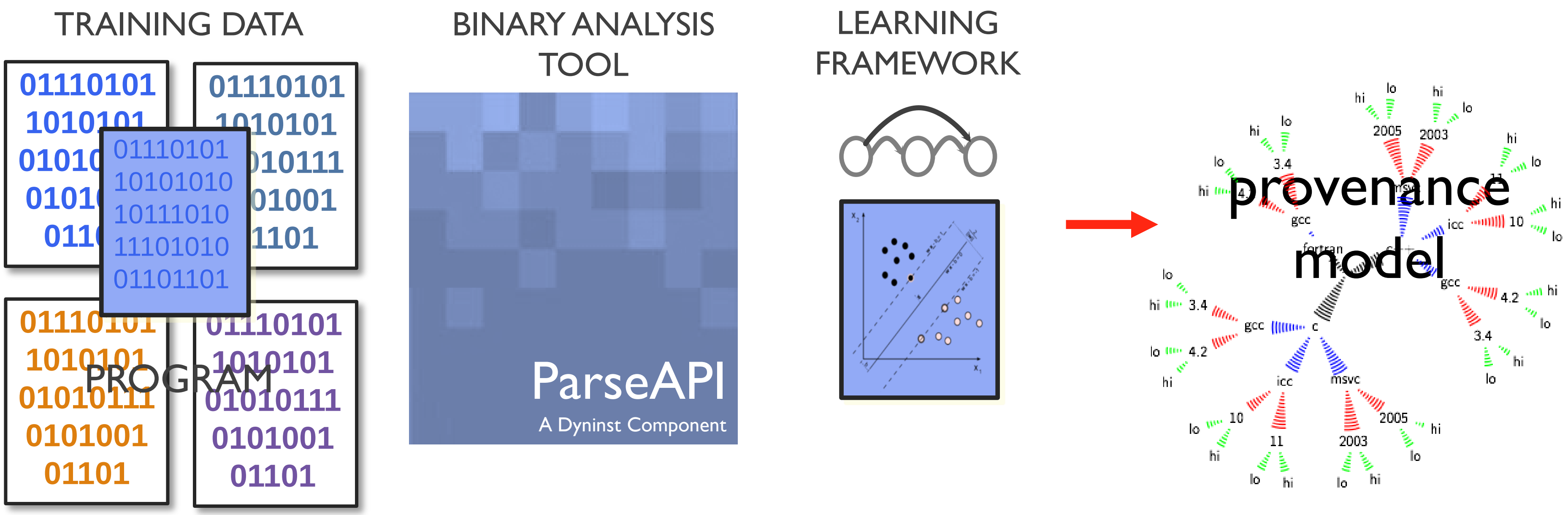
CFG of Conficker A



Binary Provenance and Authorship



Provenance System Overview



TRAINING DATA

PROGRAM

BINARY ANALYSIS TOOL

ParseAPI

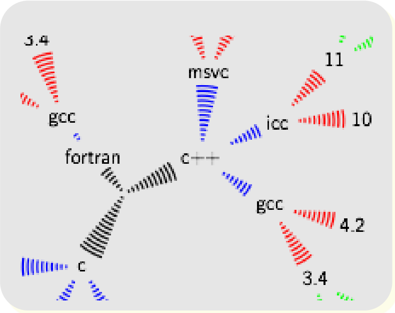
A Dyninst Component

LEARNING FRAMEWORK

provenance model

Provenance Evaluation

175 programs x



→ 2,686 binaries → 955k functions

LO HI

	Language	Compiler	Optimization	Version
Acc.	.999	.998	.993	.910

Instrumenting Virtualized Environments

Windows
Linux
Solaris
Windows
Linux
Solaris

vmware

Xen

Dyn inst

Status Update

Dyninst 7.0.1

Major new features:

- New platforms for binary rewriter
  - x86 and x86_64 - statically linked binaries
  - ppc32 and BlueGene/P - dynamically linked binaries
- Improvements to parsing speed
- Reductions in memory usage

- Deprecated Solaris and IA64 platforms
- AIX pending due to support difficulties

Component Status Update

- SymtabAPI 7.0.1
  - Speed and space optimizations
- InstructionAPI 7.0.1
  - PowerPC (ppc32, ppc64) platform
  - Full integration with Dyninst
- ParseAPI 7.0.1 - Platform independent API for parsing binaries
  - Control flow graph representation
  - Interprocedural edges (call and return)
  - Built on InstructionAPI and SymtabAPI
  - Full integration with Dyninst

Component Status Update

- StackwalkerAPI 2.1
  - Significant reduction in memory usage
- ProcControlAPI 1.0.1 - Platform independent interface for creating, monitoring and controlling processes
  - High level abstraction for process control, breakpoints and callbacks for process events
- DynC API 1.0.1 - Instrumentation language for specifying snippets
  - C like instrumentation snippets for easy and more legible mutator
  - Handles creation and destruction of snippet-local variables

Dyninst 8.0

- ProcControl API - Windows and BlueGene
- Stackwalker API - Windows and VxWorks
- Stackwalker & ProcControl integration into Dyninst

- PatchAPI and integration into Dyninst
- SR Dyninst for tamper resistant and obfuscated binaries
- New platforms for binary rewriter
  - Dynamically linked binaries on ppc64 and Windows
  - Statically linked binaries on ppc32 and BlueGene/P
- Dataflow API official release

MRNet 3.0.1

- Support for loading several filters from the same library
- Lightweight MRNet back-end support for non-blocking receives
- CrayXT support for staging files using ALPS tool helper
- Improved build structure that permits configuration for multiple platforms from a single source distribution
- Numerous bug fixes and enhancements

Software and Manuals

- Dyninst 7.0.1, MRNet 3.0.1: *available now!*
- Downloads:

  http://www.paradyn.org/html/downloads.html

  http://www.paradyn.org/html/manuals.html

- Dyninst 8.0 – 4th quarter, 2011
- MRNet 3.0.2 – coming soon!

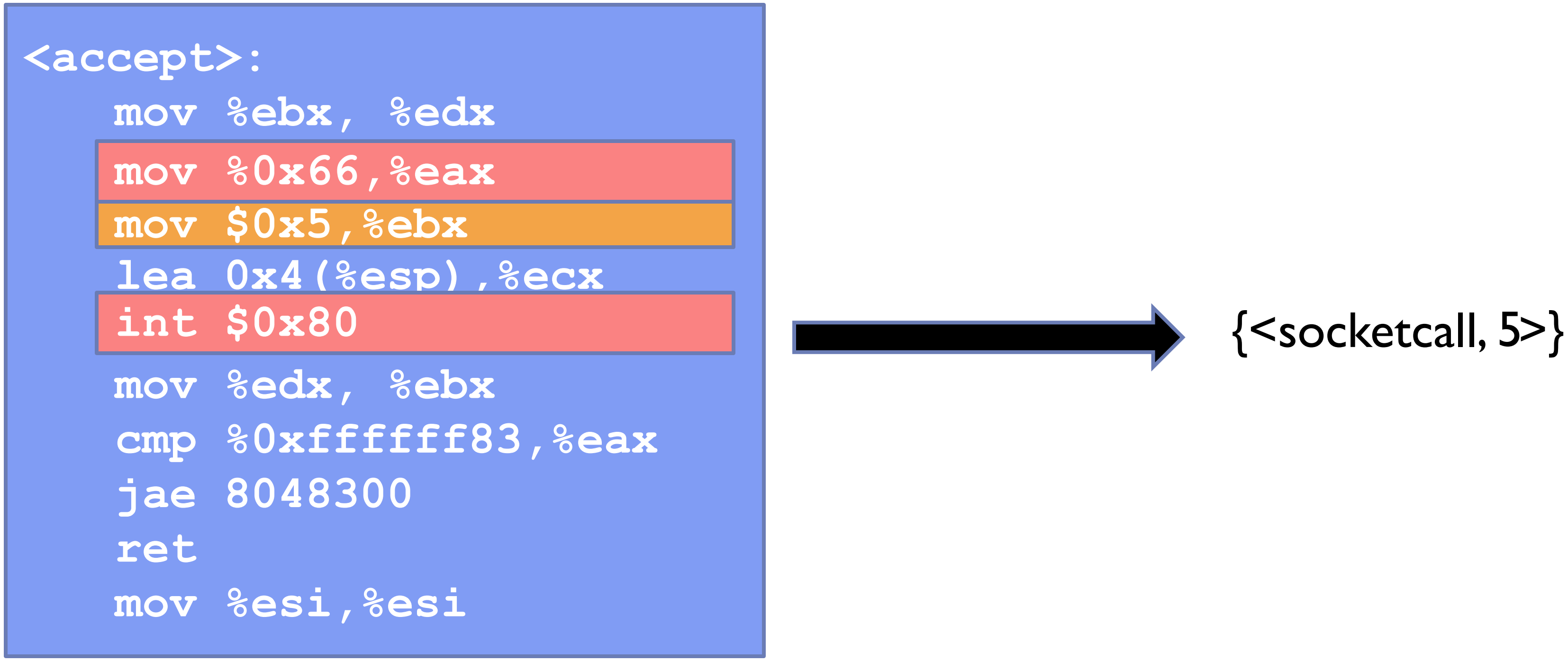
New Environments

- Virtual Machines
  - Whole-system profiling (guest + VMM) using instrumentation
  - VMM-level information to understand how and why an application's performance is affected by the virtualized environment
  - Expand performance profiling in the virtualized environment, where traditional approaches do not work or may not be sufficient
- Mobile environments – VxWorks, ARM
- GPUs

Questions

Unstrip: Semantic Descriptors

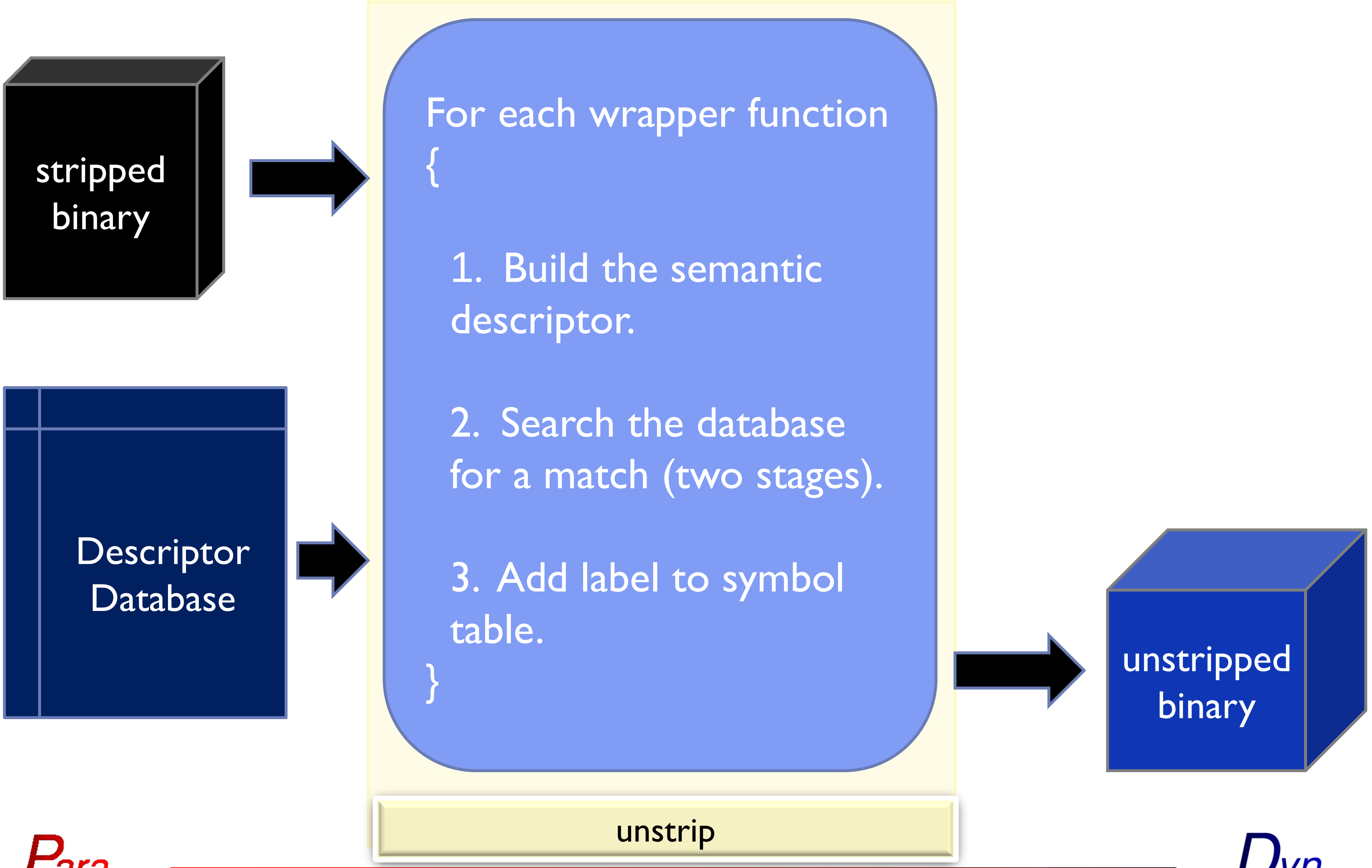
- We take a semantic approach
- Record information that is likely to be invariant across multiple versions of the function



```
<accept>:
    mov %ebx, %edx
    mov %0x66,%eax
    mov $0x5,%ebx
    lea 0x4(%esp),%ecx
    int $0x80
    mov %edx, %ebx
    cmp %0xffffff83,%eax
    jae 8048300
    ret
    mov %esi,%esi
```

{<socketcall, 5>}

Identifying Functions in a Stripped Binary



stripped binary

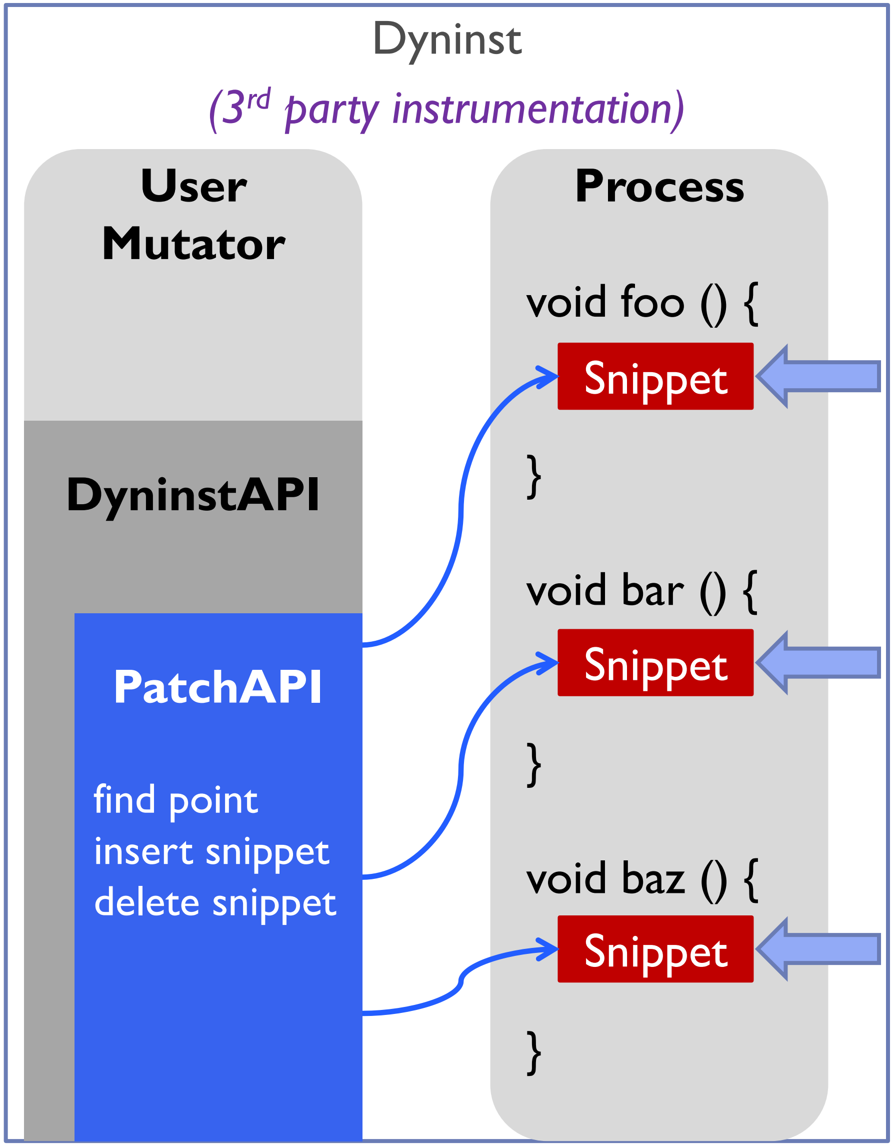
Descriptor Database

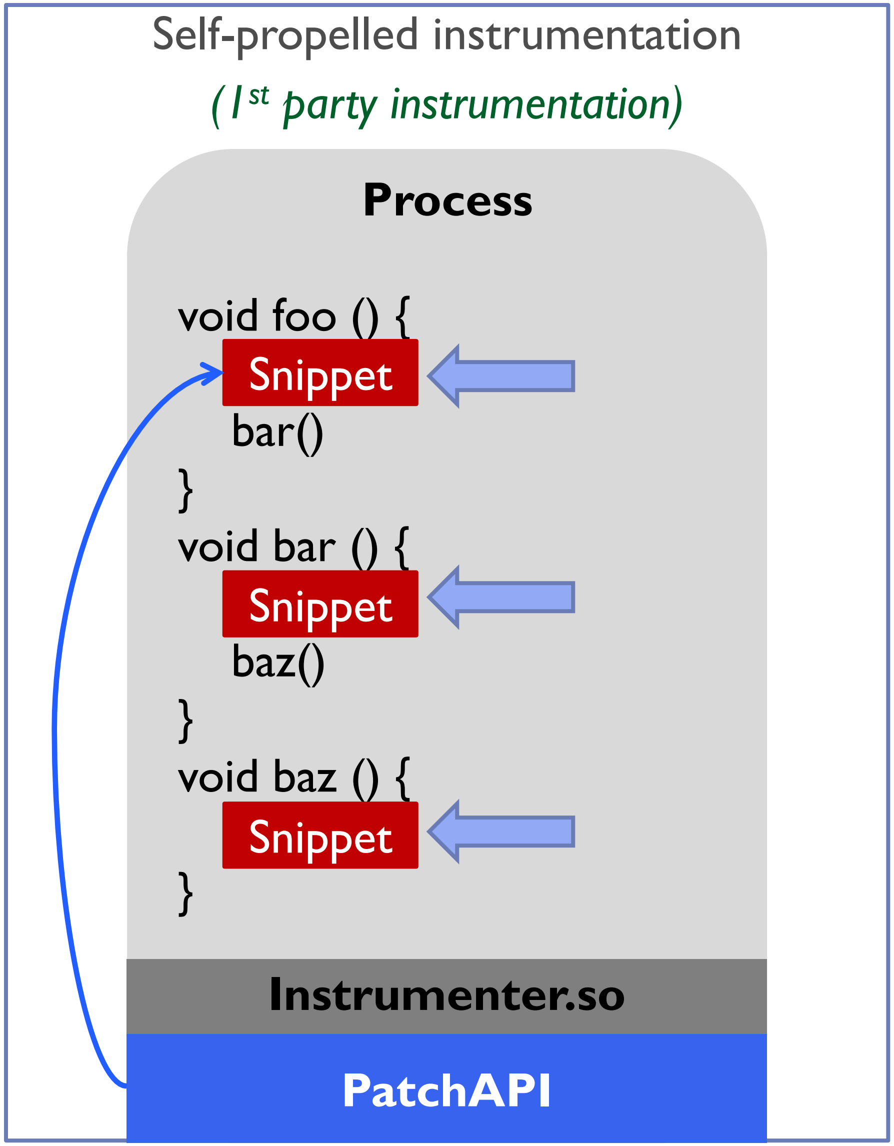
```
For each wrapper function
{

  1. Build the semantic
  descriptor.

  2. Search the database
  for a match (two stages).

  3. Add label to symbol
  table.
}
```

unstripped binary

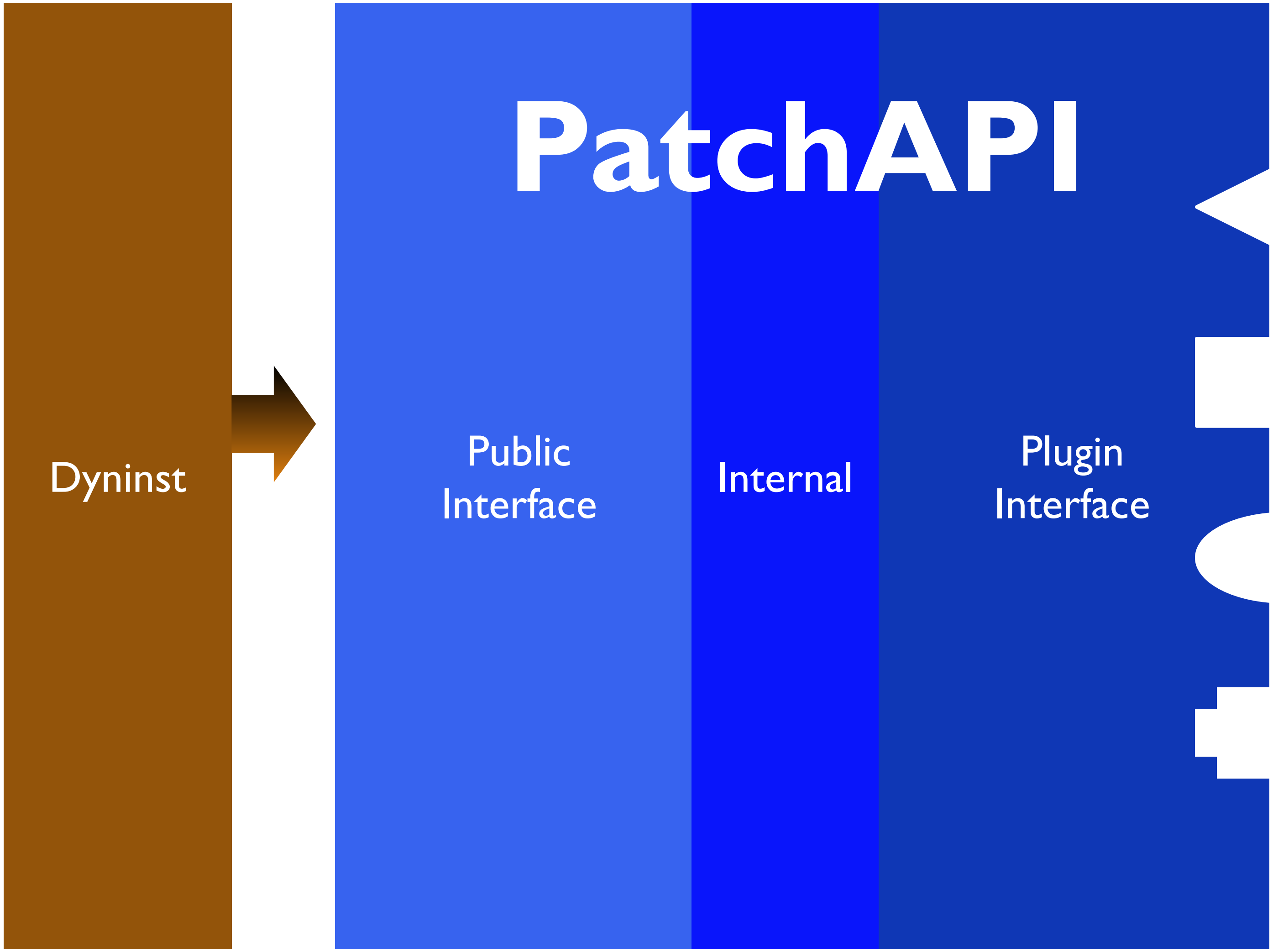
unstrip

Performance: Capturing Fine-grained behavior

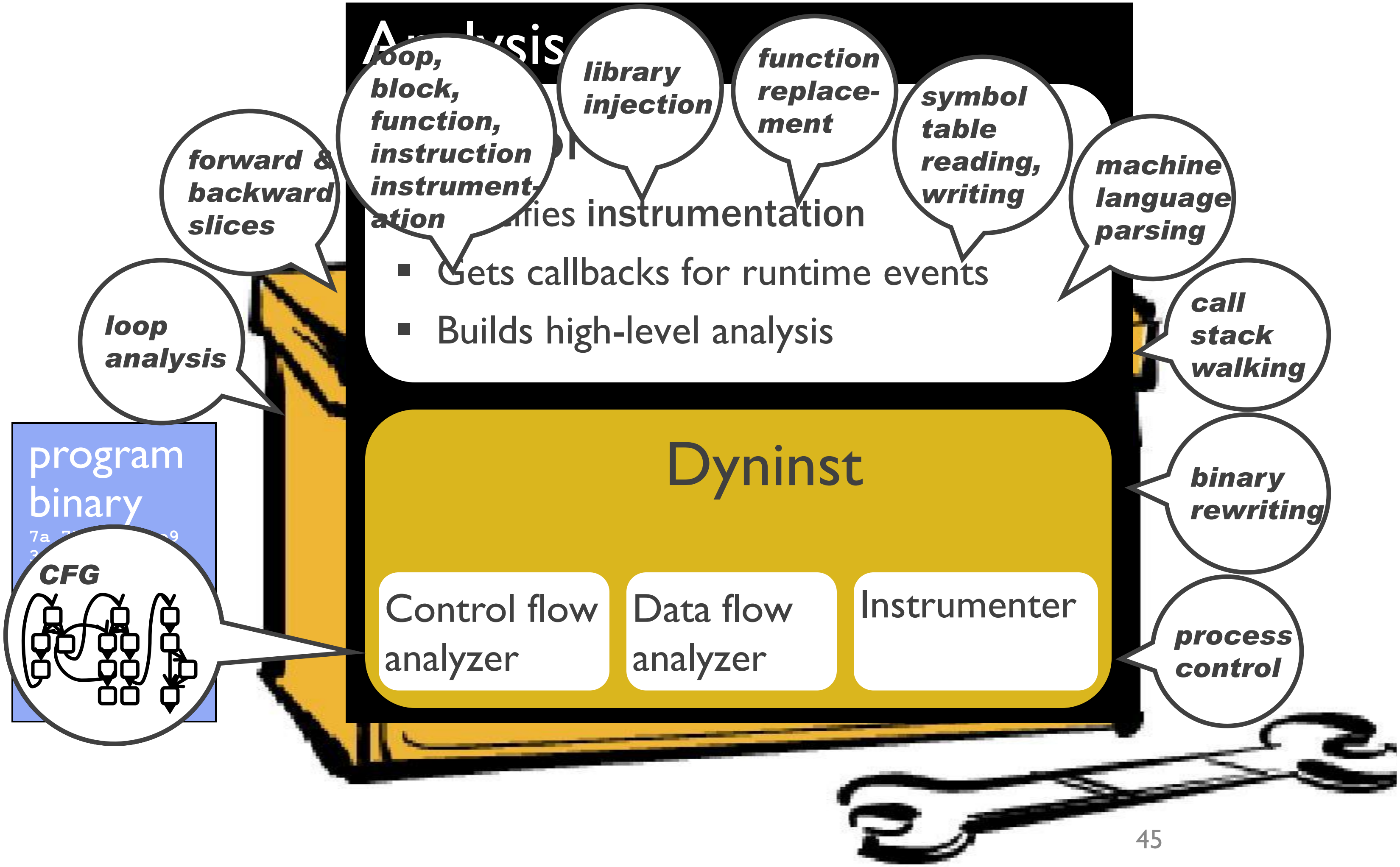




New Component: PatchAPI



Dyninst is a toolbox for analysts



What we could do because of components?

- SymtabAPI & StackwalkerAPI
- DyninstAPI Instrumentor
- ROSE semantics engine
- Tools we developed - quickly
    - Binary rewriter unstrip

Stack Trace Analysis Tool

2011 R&D 100

Open | SpeedShop™

HPCToolkit Rice University

scalasca

CRAY® THE SUPERCOMPUTER COMPANY

τ

Componentization

- Trade-offs
  - Internal requirements vs. external requirements
  - Early feedback vs. interface stability
  - Development time vs. scope
  - Structured vs. organic
- Lesson learned
  - Keep the project details where they belong
  - Change code incrementally
  - Test new interfaces

Binary rewriter

- Read binary file format from disk
- Parse binary code and build CFG
- Generate code for instrumentation
- Patch code
- Emit new binary file

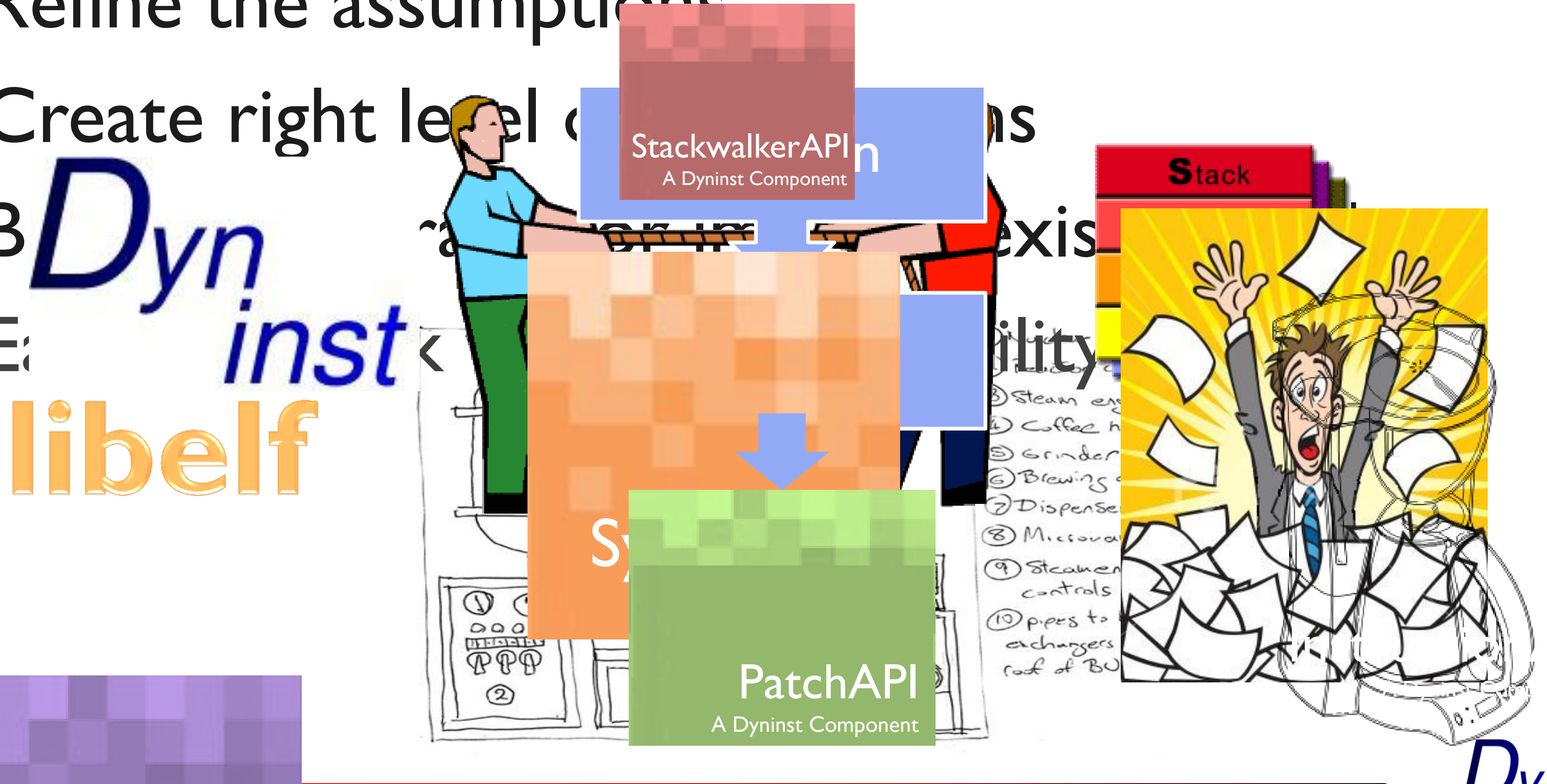
SymtabAPI ParseAPI DyninstAPI PatchAPI

Binary rewriter

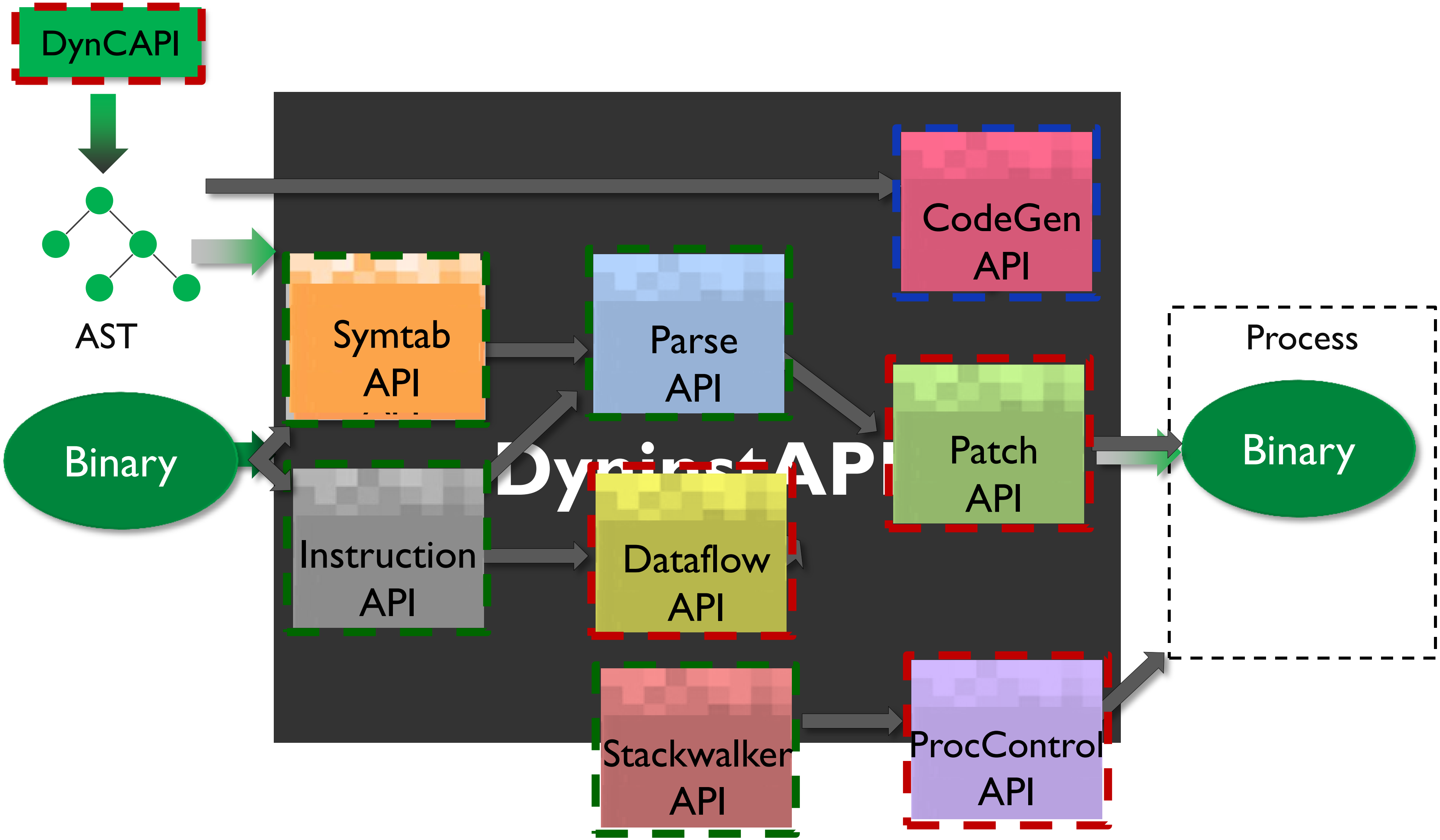
Dyn
inst
ParseAPI
A Dyninst Component
ProcControlAPI
A Dyninst Component
StackwalkerAPI
A Dyninst Component
B
The GNU Project
Debugger
PatchAPI
A Dyninst Component
DataflowAPI
A Dyninst Component
Sy
A

Componentization: Design decisions

- Define the scope of the component
- Balance internal and external user requirement
- Refine the assumpti
- Create right le el ns
- B a exis
- E k ility



Dyninst and the components


DynCAPI
AST
Binary
Symtab API
Instruction API
Parse API
Dataflow API
DyninstAPI
CodeGen API
Patch API
Stackwalker API
ProcControl API
Process
Binary