The Build / Comprehend Pipelines

R. C. Holt
M. W. Godfrey
A. J. Malton

University of Waterloo

Software – Building and Comprehension

claim:

A reverse-engineering process
applied to a code base
should mimic the build procedure
to effectively reveal
the software design

Source File Inclusion

```
COPY "CUSTFILE".
…

    MOVE MY-VAR TO CUST-START.
…
```

PROG01.CBL

Source File Inclusion

```
COPY "CUSTFILE".
…


    MOVE MY-VAR TO CUST-START.
…
```

PROG01.CBL

```
COPY "CUSTFILE".


…
    MOVE CUST-START TO MY-DATE.
    MOVE MY-YY TO CCYY.
…
```

PROG02.CBL

Source File Inclusion

```
01   CUST-REC.
…
     05 MY-VAR PIC 9(6).
…
     MOVE MY-VAR TO CUST-START.
```

PROG01.CBL

```
01   CUST-REC.
…
     05 MY-VAR PIC 9(6).
…
     MOVE CUST-START TO MY-DATE.
     MOVE MY-YY TO CCYY.
```

PROG02.CBL

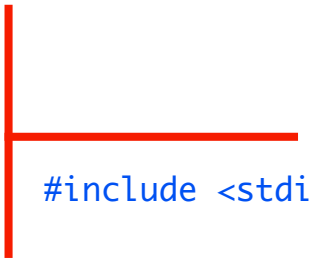
Source File Inclusion

We can't ignore

what the file-includer knows

and still comprehend the modular structure from the code.

Preprocessing



```
#include <stdio.h>
```

```
int main ()
{
     char input [100];
     fgets (input, sizeof input, stdin);
     if (feof (stdin))
          putc ('Y', stdout);
}
```

Preprocessing

```
…
extern FILE __sF [ ] ;
…
static __inline int __sputc ( int _c , FILE * _p ) {
    if ( -- _p -> _w >= 0 || ( _p -> _w >= _p -> _lbfsize && ( char ) _c != '\n' ) )
    return ( * _p -> _p ++ = _c ) ;
else
    return ( __swbuf ( _c , _p ) ) ;
}

int main ( )
{
    char input [ 100 ] ;
    fgets ( input , sizeof input , ( & __sF [ 0 ] ) ) ;
    if ( ( ( ( ( & __sF [ 0 ] ) ) -> _flags & 0x0020 ) != 0 ) )
        __sputc ( 'Y' , ( & __sF [ 0 ] ) ) ;
}
```

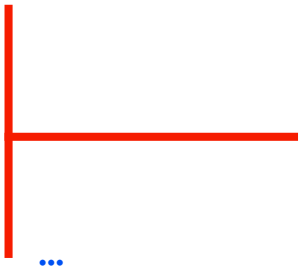
Preprocessing

We can't ignore

what the preprocessor knows

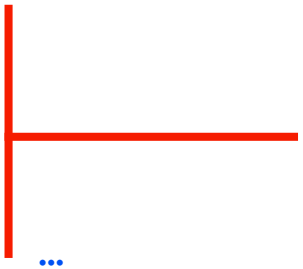
and still comprehend the definition and use of complex APIs.

Semantics



```
…
void doSomething (X &x, Y &y, Z* &z)
{
    …
    x = y;
    if (x == z)
        x = z [++ y];
    …
}
```

Semantics



```
…
void doSomething (X &x, Y &y, Z* &z)
{
    …
    x. operator = ( X::X (y) );
    if (x. operator == z)
        x. operator = (z. operator [] (y. operator ++ ());
    …
}
```

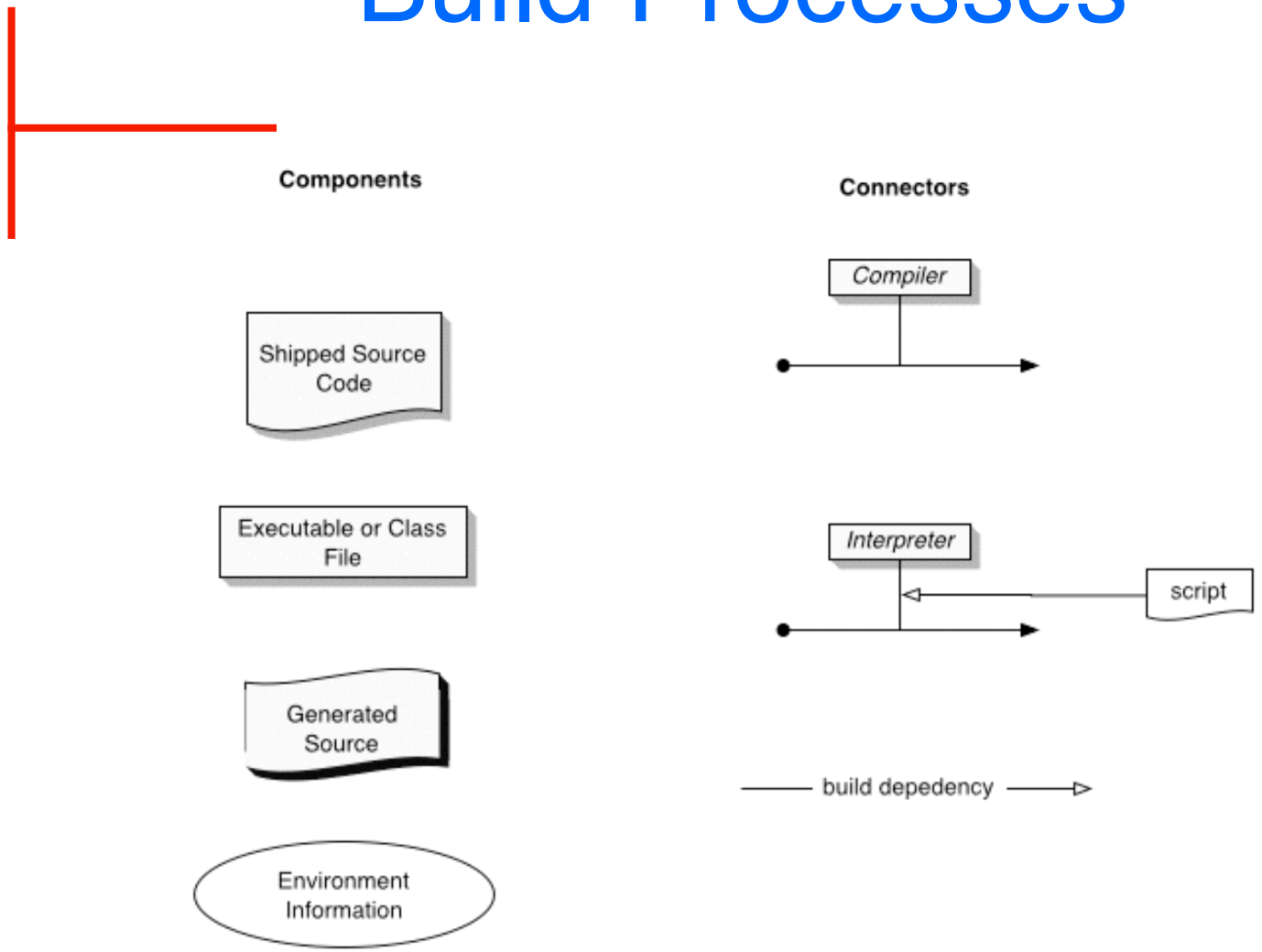
Semantics

We can’t ignore

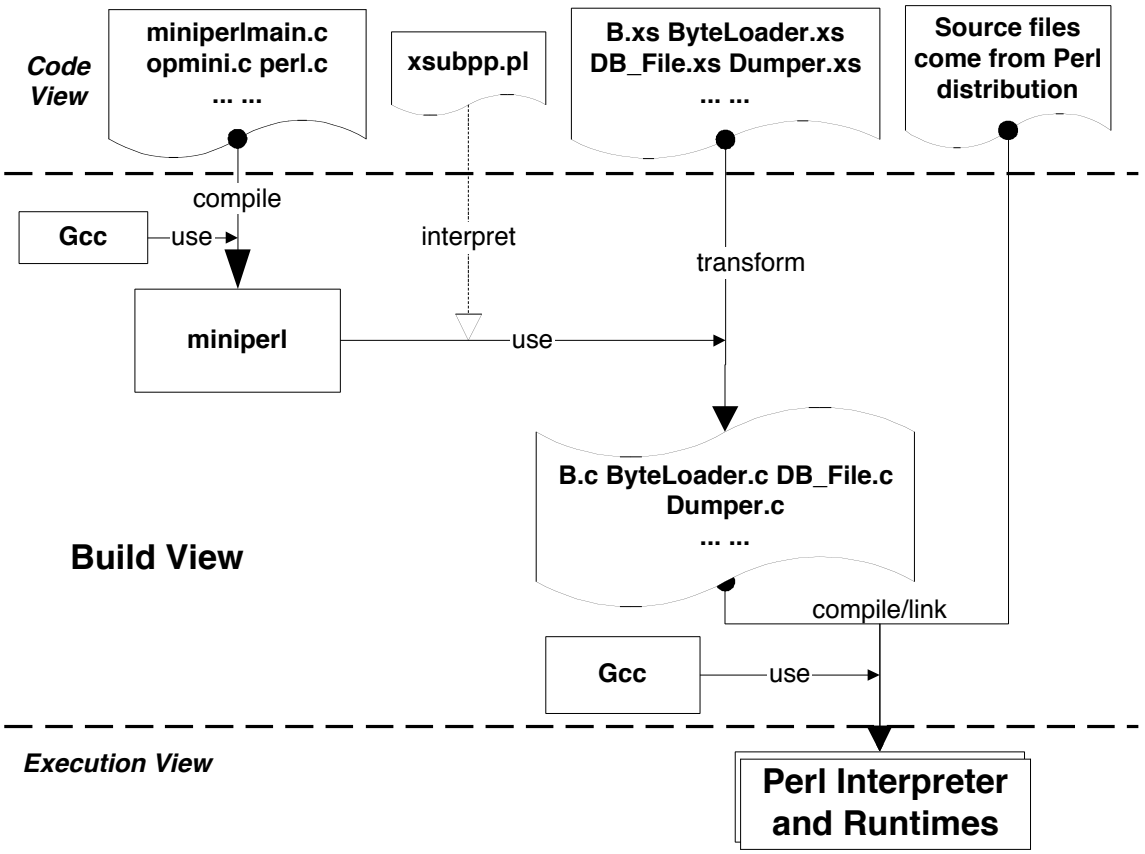
what the semantic analyser knows

by approximating based on syntax,
and even capture the call graph.

Build Processes



Complex Builds



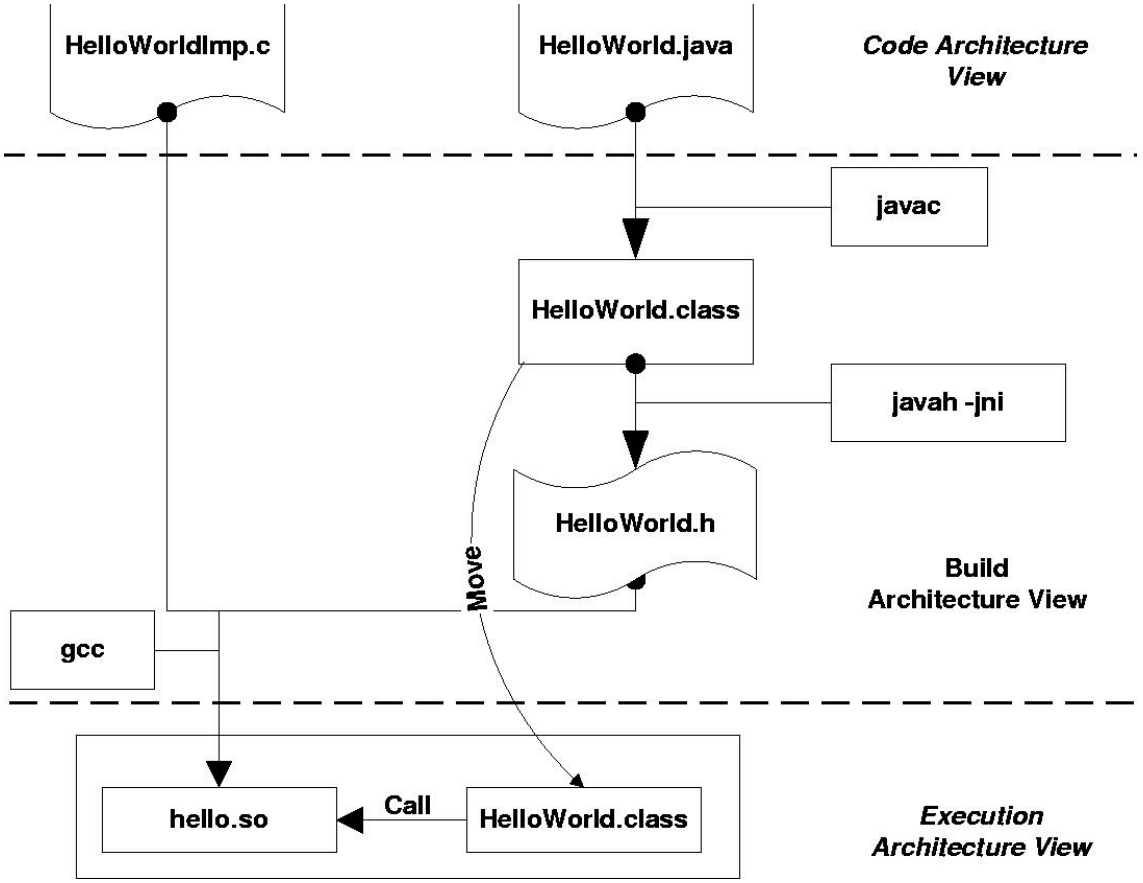
Complex Builds

We can't ignore

the dynamics of the make process

and still give a coherent picture of the architecture of bootstrapped software.

Build and Development Interleaved



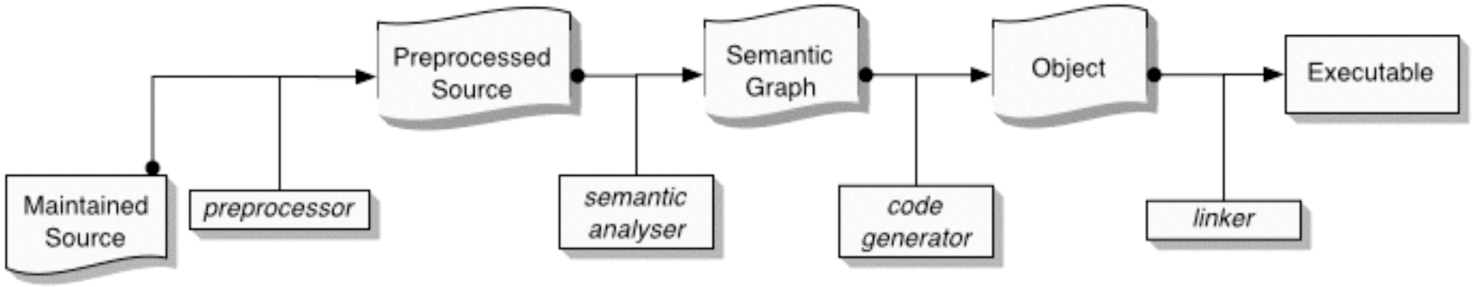
Build and Development Interleaved

We can't ignore

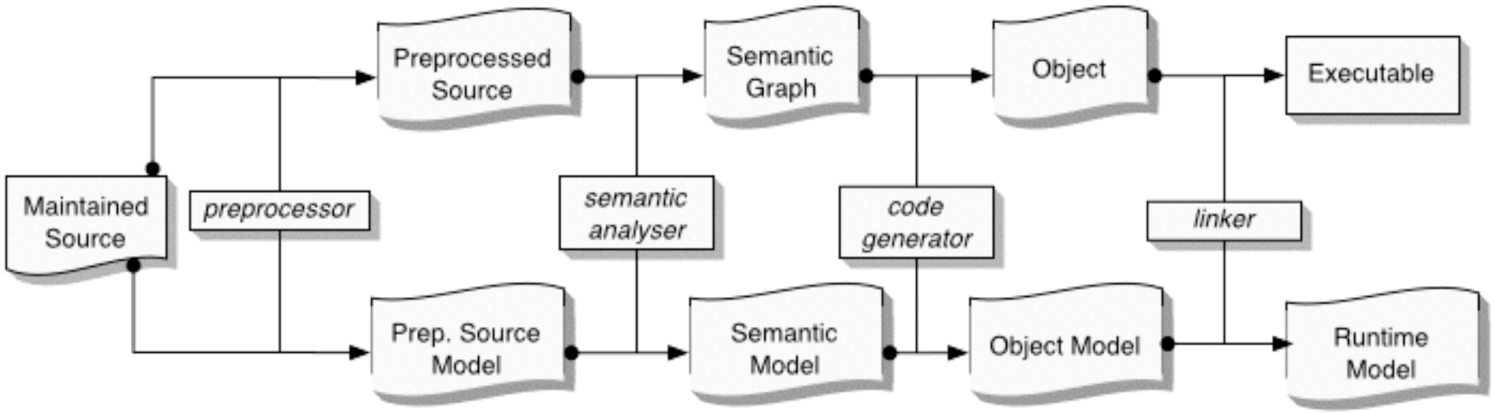
the development history

and still recognize dependencies between source and dependent artefacts.

Instrumented Build Process



Instrumented Build Process

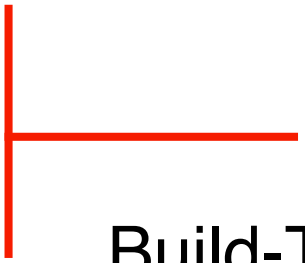


Software – Building and Comprehension

claim:

A reverse-engineering process
applied to a code base
should mimic the build procedure
to effectively reveal
the software design

Current Work



Build-Time Architectural View [Godfrey, Tu]

http://www.swag.uwaterloo.ca/~xdong/btv/

SwagKit [Holt, Bull, Malton, DuToit]

http://www.swag.uwaterloo.ca/swagkit

instrumented Make [Godfrey, Tu, Dong]

instrumented CPP [Mennie, Gonsalves]