

BERT: BEhavioral Regression Testing

Alessandro (Alex) Orso

School of CS -- College of Computing

Georgia Institute of Technology

<http://www.cc.gatech.edu/~orso/>

Tao Xie

Department of Computer Science

North Carolina State University

<http://people.engr.ncsu.edu/txie/>

2nd UPDATE: Amazon.com Web Site Down For Technical Reasons

June 06, 2008: 04:02 PM EST



(Updated to add information from a company customer-service representative.)

NEW YORK -(Dow Jones)- Amazon.com Inc.'s (AMZN) Web site was down for more than an hour Friday afternoon and remained malfunctional at the time of this report.

An Amazon.com customer-service representative said the site wouldn't be fully functional for another one or two hours. She said the outage was due to an upgrade of the company's Web site, but didn't provide further details.

The company's Web site was completely down between at least 1:40 p.m. EDT to 3 p.m. before reappearing with partial functionality.

Dow Jones Newswires employees were still unable to complete a full transaction on the site before getting an error message at the time of this report.

Sponsored Links

Options Investing Streamlined

\$9.95/Option + \$0 per Contract, Any Size. Get Flat Rate Commissions!

An investor's best friend?

In a tough market, lean on Options. The investment with many advantages.

Refinance Now at 5.2% FIXED!

\$200,000 mortgage under \$599/mo. No

2nd UPDATE: Amazon.com Web Site Down For Technical Reasons

June 06, 2008: 04:02 PM EST



(Updated to add information from a company customer-service representative.)

NEW YORK -(Dow Jones)- Amazon.com Inc.'s (AMZN) Web site was down for more than an hour Friday afternoon and remained malfunctional at the time of this report.

An Amazon.com customer-service representative said the site wouldn't be fully functional for another one or two hours. She said the outage was due to an upgrade of the company's Web site, but didn't provide further details.

The company's Web site was completely down between at least 1:40 p.m. EDT to 3 p.m. before reappearing with partial functionality.

Dow Jones Newswires employees were still unable to complete a full transaction on the site before getting an error message at the time of this report.

Sponsored Links

[Options Investing Streamlined](#)

\$9.95/Option + \$0 per Contract, Any Size. Get Flat Rate Commissions!

[An investor's best friend?](#)

In a tough market, lean on Options. The investment with many advantages.

[Refinance Now at 5.2% FIXED!](#)

\$200,000 mortgage under \$599/mo. No

2nd UPDATE: Amazon.com Web Site Down For Technical Reasons

June 06, 2008: 04:02 PM EST



(Updated to add information from a company customer-service representative.)

NEW YORK -(Dow Jones)- Amazon.com Inc.'s (AMZN) Web site was down for more than an hour Friday afternoon and remained malfunctioning.

An Amazon.com representative would not say whether the outage was caused by a software upgrade or a server problem.

The outage began at 1:40 p.m. EDT.

Dow Jones Newswires employees were still unable to complete a full transaction on the site before getting an error message at the time of this report.

[...] the outage was due to an upgrade of the company's Web site [...]

Sponsored Links

Options Investing Streamlined
\$9.95/Option + \$0 per Contract, Any Size. Get Flat Rate Commissions!

An investor's best friend?
In a tough market, lean on Options. The investment with many advantages.

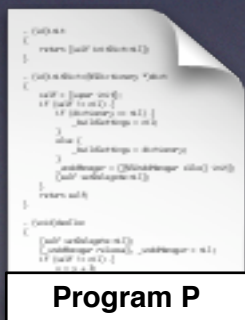
Refinance Now at 5.2% FIXED!
\$200,000 mortgage under \$599/mo. No

“If only the kernel had a
regression testsuite, everything
would be better.”

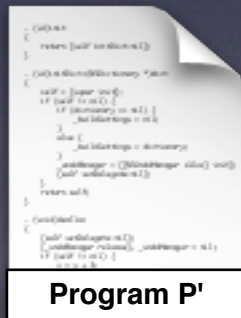
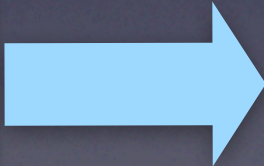
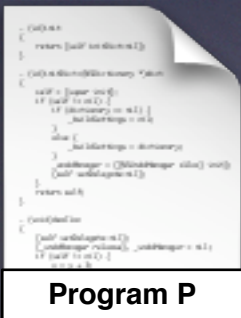
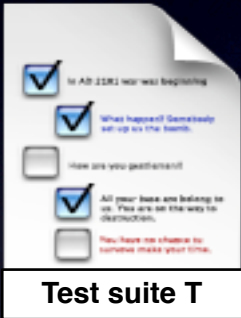


-- Greg Kroah-Hartman
keynote on the Linux
kernel at OLS 2006

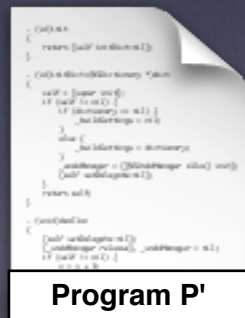
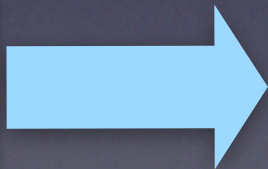
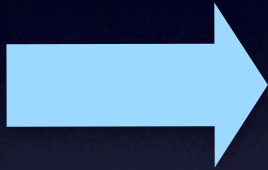
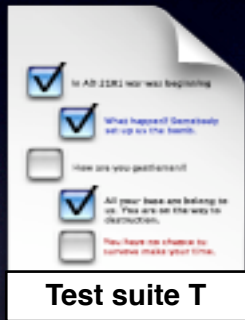
Regression Testing Process and Issues



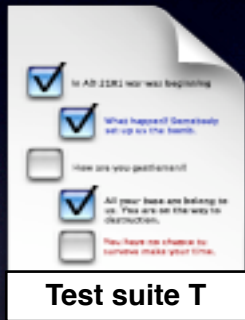
Regression Testing Process and Issues



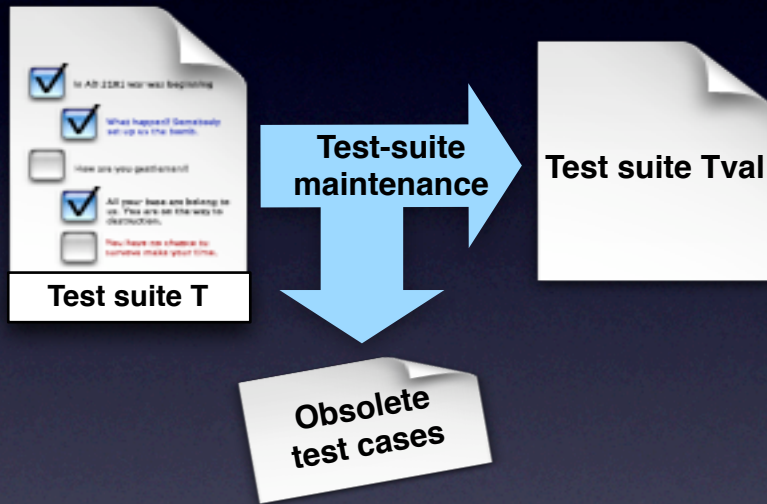
Regression Testing Process and Issues



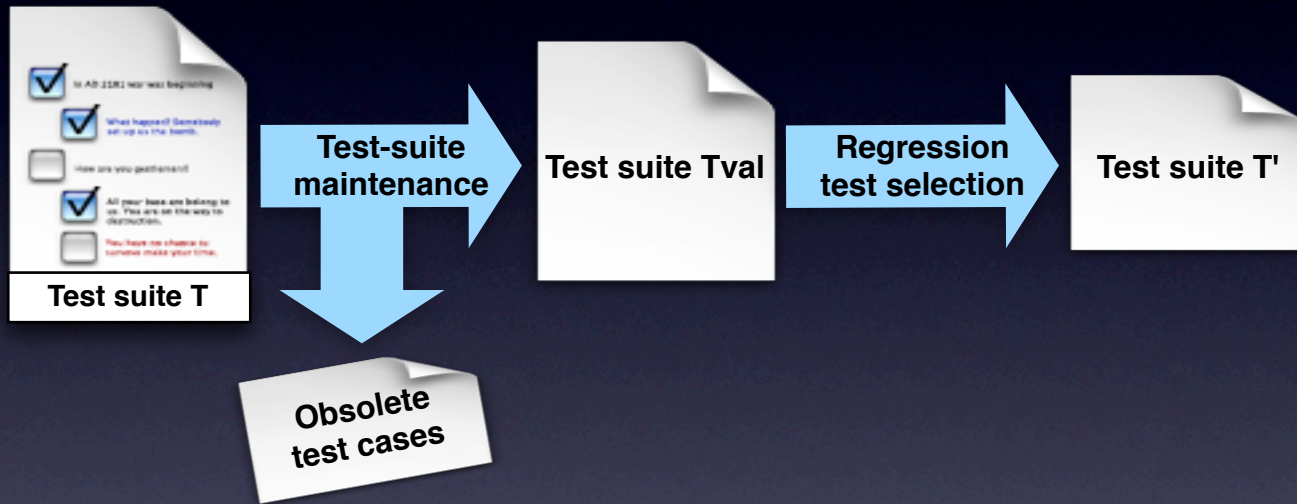
Regression Testing Process and Issues



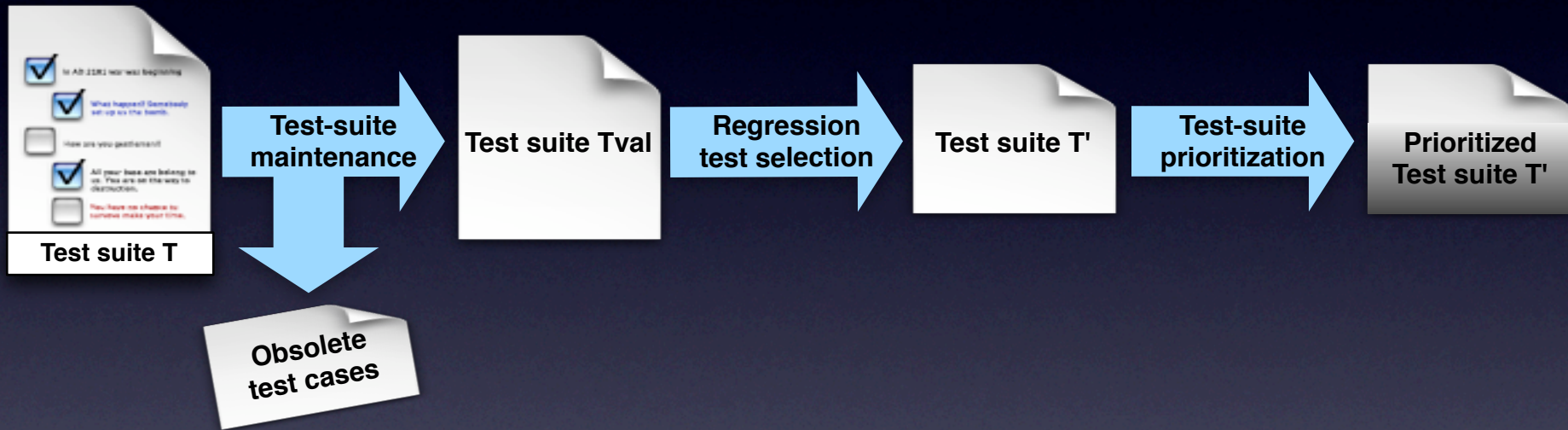
Regression Testing Process and Issues



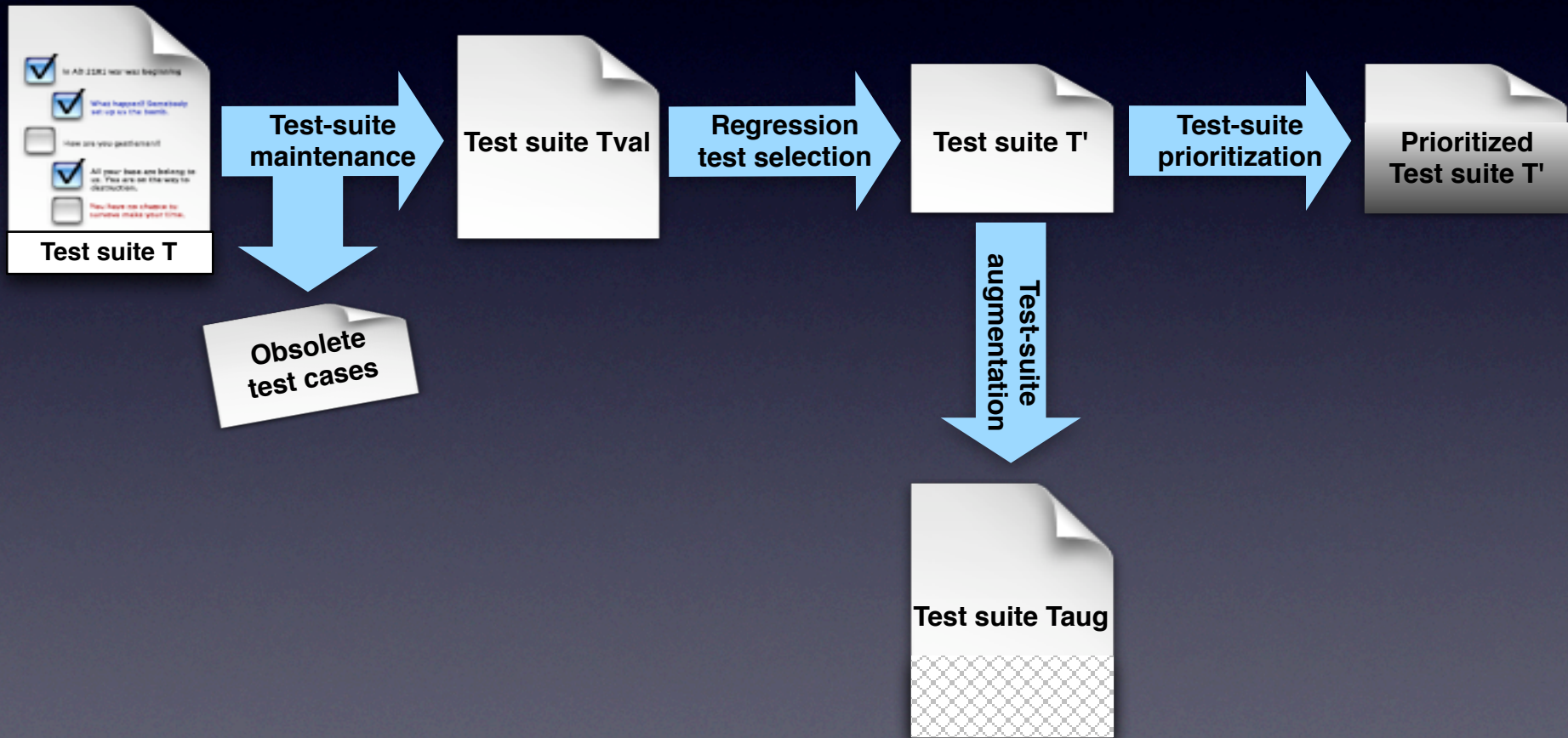
Regression Testing Process and Issues



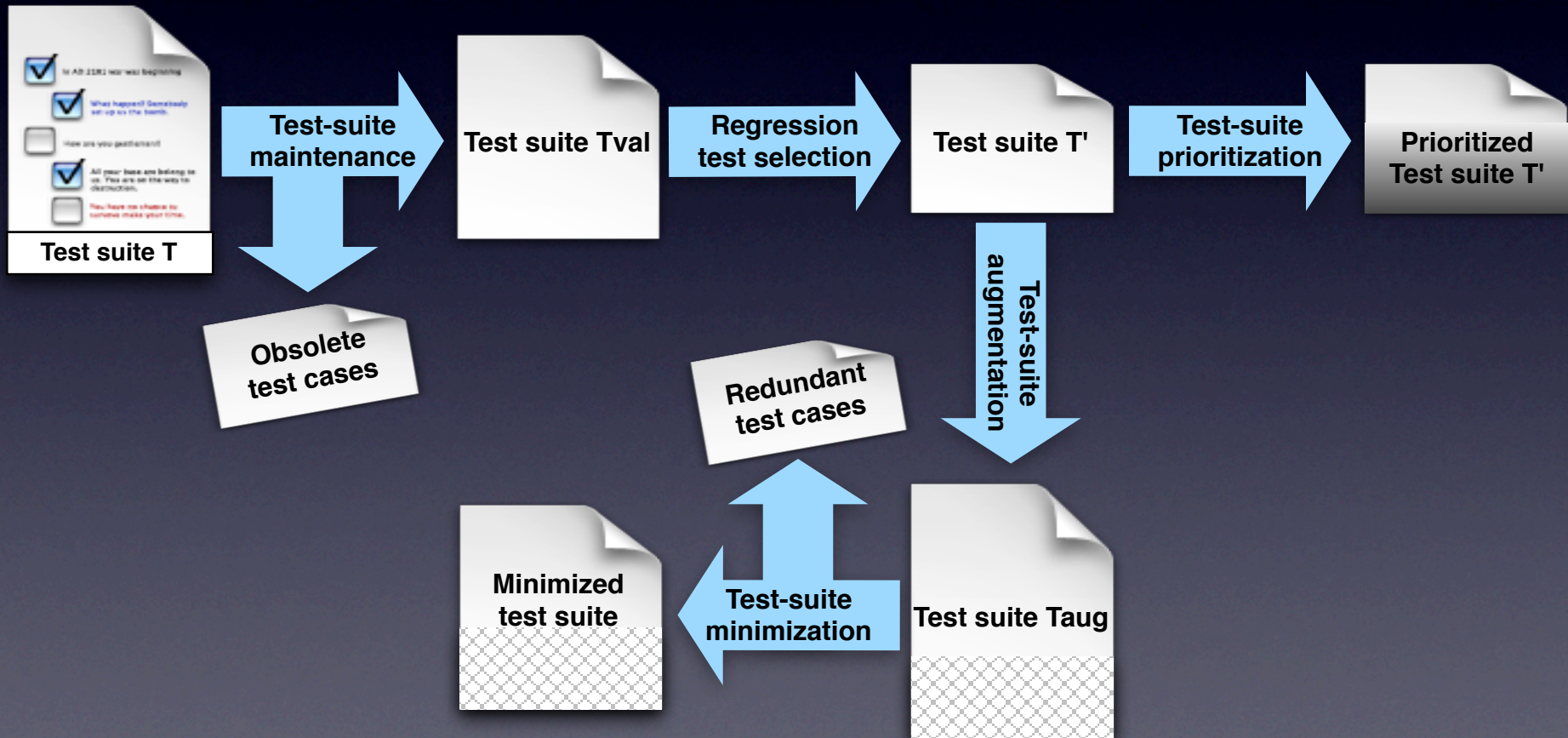
Regression Testing Process and Issues



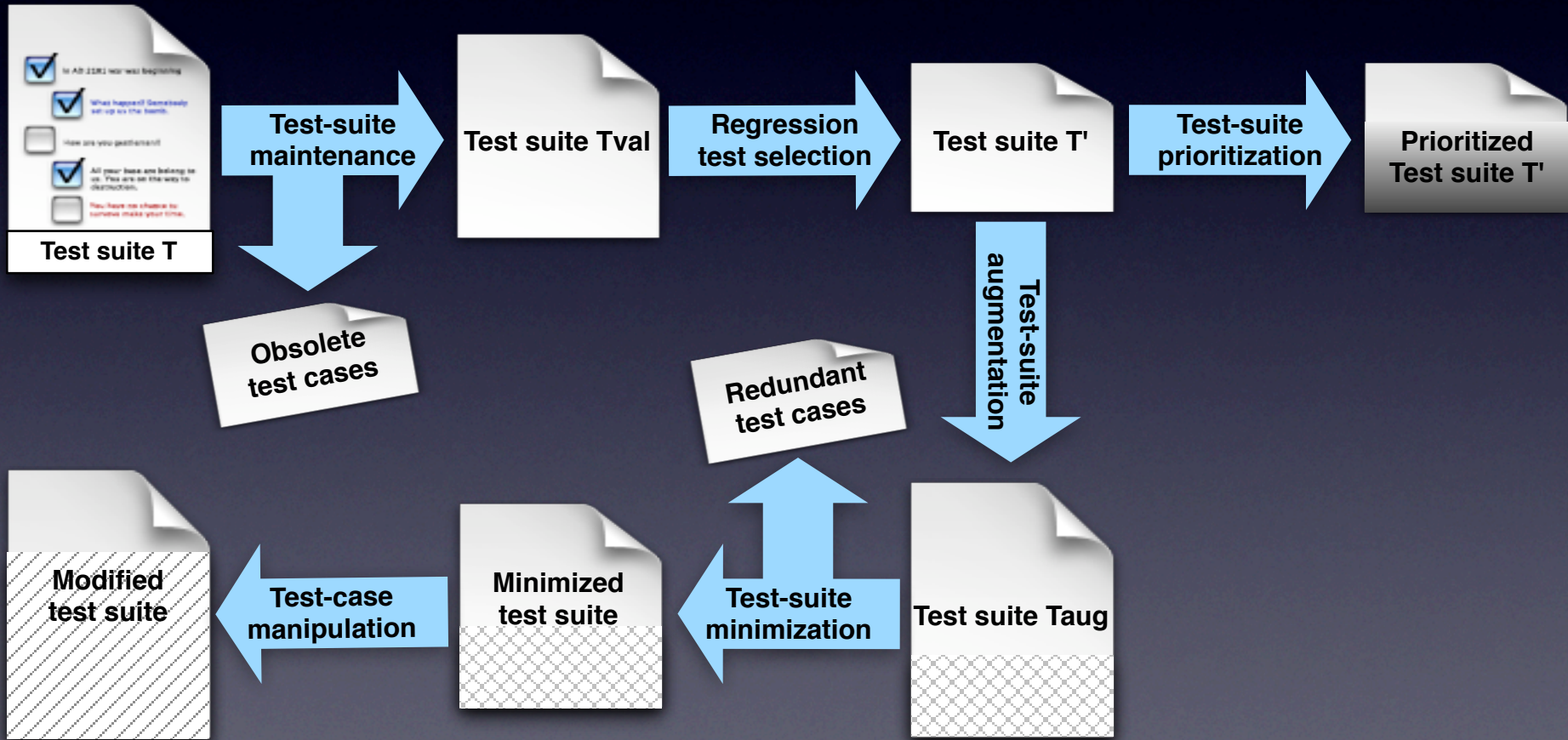
Regression Testing Process and Issues



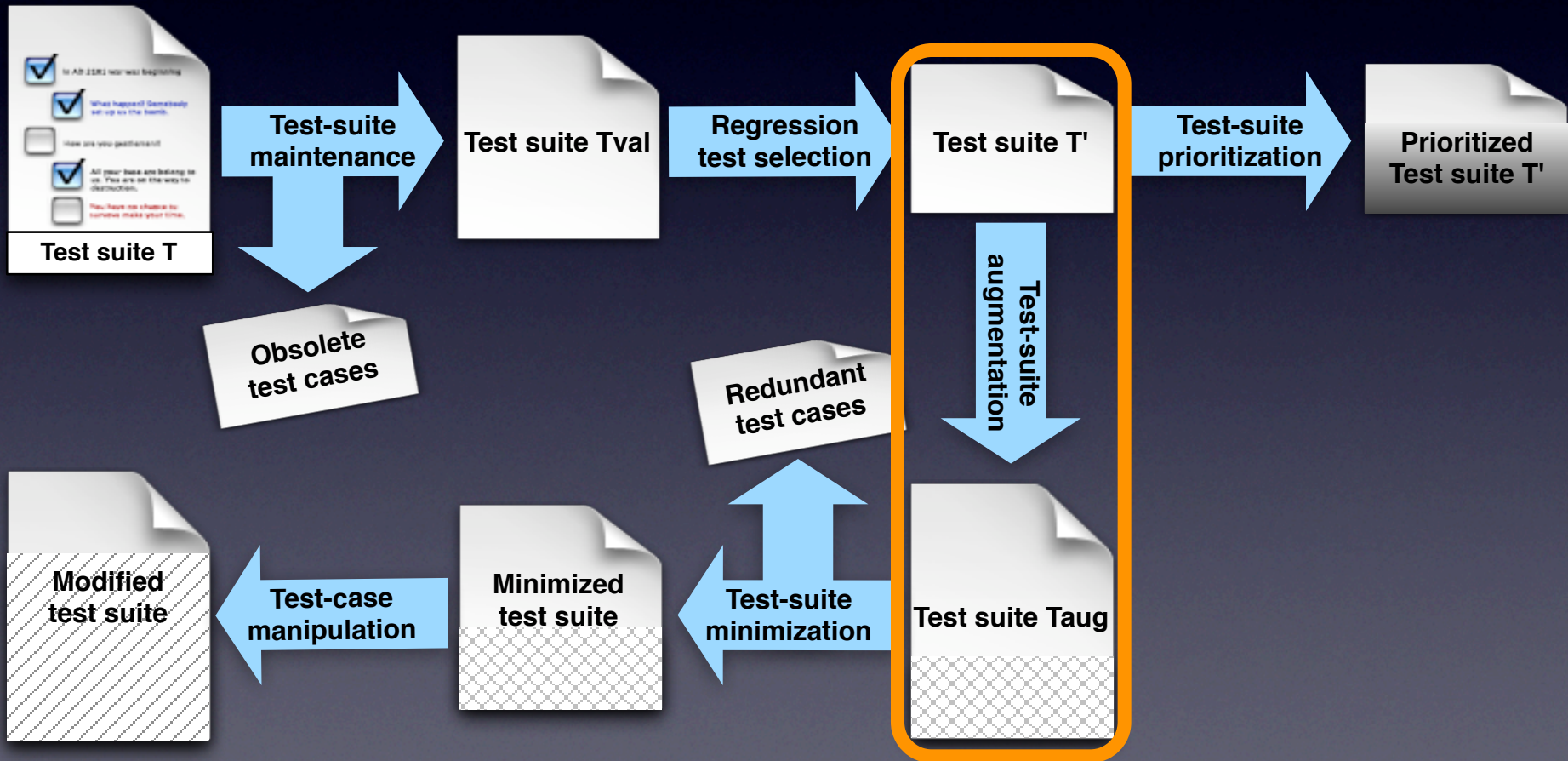
Regression Testing Process and Issues



Regression Testing Process and Issues



Regression Testing Process and Issues



Outline

- Introduction
- Our technique
- Experience
- Conclusion and future work

Outline

- Introduction
- Our technique
- Experience
- Conclusion and future work

Outline

- Introduction
- Our technique
- Experience
- Conclusion and future work

Traditional regression testing

```
- (AD) ok
1
2 return [self setUp: nil];
3
4 - (AD) ok (AD) Dictionary *item
5
6 self = [super init];
7 IF (self != nil) {
8     _adDictionary = nil;
9 }
10 else {
11     _adDictionary = dictionary;
12 }
13 _adManager = (NSMutableDictionary *) nil;
14 [self setUp: nil];
15
16 return self;
17
18 - (void) setUp
19 {
20     [self setUp: nil];
21     [_adManager removeAll];
22     [_adManager removeAll];
23     IF (self != nil) {
24         ...
25     }
26 }
```

Program P

```
- (AD) ok
1
2 return [self setUp: nil];
3
4 - (AD) ok (AD) Dictionary *item
5
6 self = [super init];
7 IF (self != nil) {
8     _adDictionary = nil;
9 }
10 else {
11     _adDictionary = dictionary;
12 }
13 _adManager = (NSMutableDictionary *) nil;
14 [self setUp: nil];
15
16 return self;
17
18 - (void) setUp
19 {
20     [self setUp: nil];
21     [_adManager removeAll];
22     [_adManager removeAll];
23     IF (self != nil) {
24         ...
25     }
26 }
```

Program P'



Test suite T

Traditional regression testing

```
- (void)run  
{  
    return [self setUpAndRun];  
}  
- (void)setUpAndRun {  
    self = [super setUp];  
    if ([self respondsToSelector:  
        @selector(setUpAndRun)]) {  
        [self setUpAndRun];  
    }  
    [self setUp];  
    [self setUpAndRun];  
    [self tearDown];  
    return self;  
}  
- (void)tearDown {  
    [self tearDown];  
    [self tearDown];  
    if ([self respondsToSelector:  
        @selector(setUpAndRun)]) {  
        [self setUpAndRun];  
    }  
}
```

Program P

```
- (void)run  
{  
    return [self setUpAndRun];  
}  
- (void)setUpAndRun {  
    self = [super setUp];  
    if ([self respondsToSelector:  
        @selector(setUpAndRun)]) {  
        [self setUpAndRun];  
    }  
    [self setUp];  
    [self setUpAndRun];  
    [self tearDown];  
    return self;  
}  
- (void)tearDown {  
    [self tearDown];  
    [self tearDown];  
    if ([self respondsToSelector:  
        @selector(setUpAndRun)]) {  
        [self setUpAndRun];  
    }  
}
```

Program P'



Test suite T

**Test runner
&
Oracle
checker**

Traditional regression testing

```
- (defn a
  1
  2 returns [self to do with it]
  3
  4)
- (defn a-b [to do with it]
  1
  2 self = [user to do]
  3 if (self to do) [
  4   _do something = a
  5 ]
  6 else [
  7   _do something = something
  8 ]
  9 _addInteger = (@StringBuilder class) new()
  10 _self _addInteger a
  11
  12 returns self
  13)
- (defn test a
  1
  2 [self _addInteger a]
  3 _addInteger "hello", _addInteger = a
  4 if (self to do) [
  5   a b c d
  6 ]
  7 )
```

Program P

```
- (defn a
  1
  2 returns [self to do with it]
  3
  4)
- (defn a-b [to do with it]
  1
  2 self = [user to do]
  3 if (self to do) [
  4   _do something = a
  5 ]
  6 else [
  7   _do something = something
  8 ]
  9 _addInteger = (@StringBuilder class) new()
  10 _self _addInteger a
  11
  12 returns self
  13)
- (defn test
  1
  2 [self _addInteger a]
  3 _addInteger "hello", _addInteger = a
  4 if (self to do) [
  5   a b c d
  6 ]
  7 )
```

Program P'

- In AD 2281 we will begining
- What happens? Somebody set up to the bomb.
- How are you getting on?
- All your base are belong to us. You are on the way to destruction.
- You have no choice to survive make your time.

Test suite T

Test runner & Oracle checker

```
There were no exceptions at
This java and it is the
app and public variable of the
the talking.
All said that. Then to it then it came
digital to the the same time as the
the talking. This able to reproduce it
in the field of the the same time
and change the situation you heard that
the best way to go on now.
Give a list, and it is the same as the
it will be the same as the same as the
to be the same as the same as the
```

Regression errors

Traditional regression testing

```
class Account {
public:
    Account(int money) {
        m = money;
    }
    Account(int money, string name) {
        m = money;
        n = name;
    }
    void setMoney(int money) {
        m = money;
    }
    void setName(string name) {
        n = name;
    }
    int getMoney() const {
        return m;
    }
    string getName() const {
        return n;
    }
};
```

Program P

```
class Account {
public:
    Account(int money) {
        m = money;
    }
    Account(int money, string name) {
        m = money;
        n = name;
    }
    void setMoney(int money) {
        m = money;
    }
    void setName(string name) {
        n = name;
    }
    int getMoney() const {
        return m;
    }
    string getName() const {
        return n;
    }
};
```

Program P'

In AD 1181 we will begining

What happens? Somebody set up as the bank.

How are you getting on?

All your bank are belong to us. You are on the way to destruction.

You have no chance to survive make your time.

Test suite T

Test runner & Oracle checker

There were in the year 1181 AD. This year we will begining. We are going to make a bank. We are talking.

All we had then. We were in the year 1181 AD. This year we will begining. We are going to make a bank. We are talking.

Now we have a chance to survive make your time.

Regression errors

```
class BankAccount {
    double balance;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (balance < 0)
            print("account overdraft");
        return false;
    }
    balance = balance - amount;

    return true;
}
```

```
class BankAccount {
    double balance;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (balance < 0)
            print("account overdraft");
        return false;
    }
    balance = balance - amount;

    return true;
}
}
```



```
class BankAccount {
    double balance;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (balance < 0)
            print("account overdraft");
        return false;
    }
    balance = balance - amount;

    return true;
}
}
```

```
class BankAccount {
    double balance;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (balance < 0)
            print("account overdraft");
        return false;
    }
    balance = balance - amount;

    return true;
}
}
```

```
class BankAccount {
    double balance;
    bool isOverdraft;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (isOverdraft) {
            print("account overdraft");
            return false;
        }
        balance = balance - amount;
        if (balance < 0)
            isOverdraft = true;
        return true;
    }
}
```

```
class BankAccount {
    double balance;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (balance < 0)
            print("account overdraft");
        return false;
    }
    balance = balance - amount;

    return true;
}
}
```

```
class BankAccount {
    double balance;
    bool isOverdraft;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (isOverdraft) {
            print("account overdraft");
            return false;
        }
        balance = balance - amount;
        if (balance < 0)
            isOverdraft = true;
        return true;
    }
}
```

```
class BankAccount {
    double balance;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (balance < 0)
            print("account overdraft");
        return false;
    }
    balance = balance - amount;

    return true;
}
}
```

```
class BankAccount {
    double balance;
    bool isOverdraft;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (isOverdraft) {
            print("account overdraft");
            return false;
        }
        balance = balance - amount;
        if (balance < 0)
            isOverdraft = true;
        return true;
    }
}
```

```
class BankAccount {
    double balance;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (balance < 0)
            print("account overdraft");
        return false;
    }
    balance = balance - amount;

    return true;
}
}
```

```
class BankAccount {
    double balance;
    bool isOverdraft;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (isOverdraft) {
            print("account overdraft");
            return false;
        }
        balance = balance - amount;
        if (balance < 0)
            isOverdraft = true;
        return true;
    }
}
```

Where is
the fault?

```
class BankAccount {  
    double balance;  
    bool isOverdraft;  
  
    bool deposit(double amount) {  
        if (amount > 0.00) {  
            balance = balance + amount;  
            return true;  
        } else {  
            print("negative amount");  
            return false;  
        }  
    }  
  
    bool withdraw(double amount) {  
        if (amount <= 0) {  
            print("negative amount");  
            return false;  
        }  
        if (isOverdraft) {  
            print("account overdraft");  
            return false;  
        }  
        balance = balance - amount;  
        if (balance < 0)  
            isOverdraft = true;  
        return true;  
    }  
}
```

```
class BankAccount {  
    double balance;  
    bool isOverdraft;  
  
    bool deposit(double amount) {  
        if (amount > 0.00) {  
            balance = balance + amount;  
            return true;  
        } else {  
            print("negative amount");  
            return false;  
        }  
    }  
  
    bool withdraw(double amount) {  
        if (amount <= 0) {  
            print("negative amount");  
            return false;  
        }  
        if (isOverdraft) {  
            print("account overdraft");  
            return false;  
        }  
        balance = balance - amount;  
        if (balance < 0)  
            isOverdraft = true;  
        return true;  
    }  
}
```

```
...
void testBehavioralDifference() {
    BankAccount a =
        new BankAccount();
    a.deposit(10.00);
    a.withdraw(20.00);
    a.deposit(50.00);
    bool result = a.withdraw(20.00);
    assertEquals(result, true);
}
...
```

```
class BankAccount {
    double balance;
    bool isOverdraft;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (isOverdraft) {
            print("account overdraft");
            return false;
        }
        balance = balance - amount;
        if (balance < 0)
            isOverdraft = true;
        return true;
    }
}
```



```

...
void testBehavioralDifference() {
    BankAccount a =
        new BankAccount();
    a.deposit(10.00);
    a.withdraw(20.00);
    a.deposit(50.00);
    bool result = a.withdraw(20.00);
    assertEquals(result, true);
}
...

```

- Such a test may not be in T
 - 100% stmt coverage without it
 - Specific sequence of calls/params
- Or its oracle may be inadequate

```

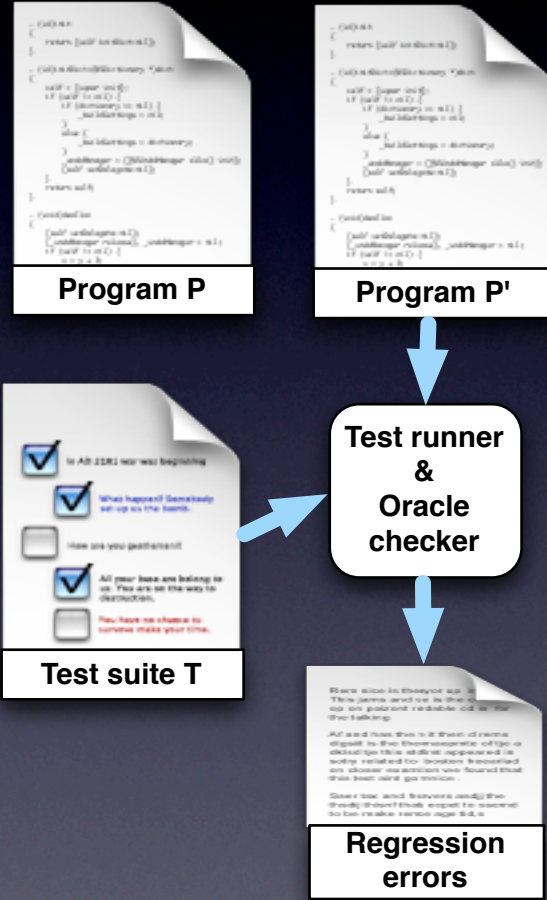
class BankAccount {
    double balance;
    bool isOverdraft;

    bool deposit(double amount) {
        if (amount > 0.00) {
            balance = balance + amount;
            return true;
        } else {
            print("negative amount");
            return false;
        }
    }

    bool withdraw(double amount) {
        if (amount <= 0) {
            print("negative amount");
            return false;
        }
        if (isOverdraft) {
            print("account overdraft");
            return false;
        }
        balance = balance - amount;
        if (balance < 0)
            isOverdraft = true;
        return true;
    }
}

```

Traditional regression testing



Existing test suite typically targets a small subset of the program behavior

- Tests focus on core functionality
- Oracles often approximated

BERT

```
def m:  
    return [self.m_embeddings]  
}  
  
def m_embeddings(self, tokens):  
    self.m = [token.m() for token in tokens]  
    if self.m is None:  
        self.m_embeddings = None  
    else:  
        self.m_embeddings = torch.cat(self.m_embeddings, self.m)  
    return self.m_embeddings  
}  
  
def m_embeddings(self):  
    return self.m_embeddings
```

Program P

```
def m:  
    return [self.m_embeddings]  
}  
  
def m_embeddings(self, tokens):  
    self.m = [token.m() for token in tokens]  
    if self.m is None:  
        self.m_embeddings = None  
    else:  
        self.m_embeddings = torch.cat(self.m_embeddings, self.m)  
    return self.m_embeddings  
}  
  
def m_embeddings(self):  
    return self.m_embeddings
```

Program P'

In AD 2181 we will begining

What happen? Somebody set up in the beach.

How are you pasternent?

All your base are belong to us. You are on the way to destruction.

You have no choice to control make your time.

Test suite T

BERT

Phase I:
Generation of
test cases for
changed code

Change
analyzer

```
def m1():  
    return [self.m1(), self.m2()]  
  
def m2():  
    return [self.m3(), self.m4()]  
  
def m3():  
    return [self.m5(), self.m6()]  
  
def m4():  
    return [self.m7(), self.m8()]  
  
def m5():  
    return [self.m9(), self.m10()]  
  
def m6():  
    return [self.m11(), self.m12()]  
  
def m7():  
    return [self.m13(), self.m14()]  
  
def m8():  
    return [self.m15(), self.m16()]  
  
def m9():  
    return [self.m17(), self.m18()]  
  
def m10():  
    return [self.m19(), self.m20()]  
  
def m11():  
    return [self.m21(), self.m22()]  
  
def m12():  
    return [self.m23(), self.m24()]  
  
def m13():  
    return [self.m25(), self.m26()]  
  
def m14():  
    return [self.m27(), self.m28()]  
  
def m15():  
    return [self.m29(), self.m30()]  
  
def m16():  
    return [self.m31(), self.m32()]  
  
def m17():  
    return [self.m33(), self.m34()]  
  
def m18():  
    return [self.m35(), self.m36()]  
  
def m19():  
    return [self.m37(), self.m38()]  
  
def m20():  
    return [self.m39(), self.m40()]  
  
def m21():  
    return [self.m41(), self.m42()]  
  
def m22():  
    return [self.m43(), self.m44()]  
  
def m23():  
    return [self.m45(), self.m46()]  
  
def m24():  
    return [self.m47(), self.m48()]  
  
def m25():  
    return [self.m49(), self.m50()]  
  
def m26():  
    return [self.m51(), self.m52()]  
  
def m27():  
    return [self.m53(), self.m54()]  
  
def m28():  
    return [self.m55(), self.m56()]  
  
def m29():  
    return [self.m57(), self.m58()]  
  
def m30():  
    return [self.m59(), self.m60()]  
  
def m31():  
    return [self.m61(), self.m62()]  
  
def m32():  
    return [self.m63(), self.m64()]  
  
def m33():  
    return [self.m65(), self.m66()]  
  
def m34():  
    return [self.m67(), self.m68()]  
  
def m35():  
    return [self.m69(), self.m70()]  
  
def m36():  
    return [self.m71(), self.m72()]  
  
def m37():  
    return [self.m73(), self.m74()]  
  
def m38():  
    return [self.m75(), self.m76()]  
  
def m39():  
    return [self.m77(), self.m78()]  
  
def m40():  
    return [self.m79(), self.m80()]  
  
def m41():  
    return [self.m81(), self.m82()]  
  
def m42():  
    return [self.m83(), self.m84()]  
  
def m43():  
    return [self.m85(), self.m86()]  
  
def m44():  
    return [self.m87(), self.m88()]  
  
def m45():  
    return [self.m89(), self.m90()]  
  
def m46():  
    return [self.m91(), self.m92()]  
  
def m47():  
    return [self.m93(), self.m94()]  
  
def m48():  
    return [self.m95(), self.m96()]  
  
def m49():  
    return [self.m97(), self.m98()]  
  
def m50():  
    return [self.m99(), self.m100()]
```

Program P

```
def m1():  
    return [self.m1(), self.m2()]  
  
def m2():  
    return [self.m3(), self.m4()]  
  
def m3():  
    return [self.m5(), self.m6()]  
  
def m4():  
    return [self.m7(), self.m8()]  
  
def m5():  
    return [self.m9(), self.m10()]  
  
def m6():  
    return [self.m11(), self.m12()]  
  
def m7():  
    return [self.m13(), self.m14()]  
  
def m8():  
    return [self.m15(), self.m16()]  
  
def m9():  
    return [self.m17(), self.m18()]  
  
def m10():  
    return [self.m19(), self.m20()]  
  
def m11():  
    return [self.m21(), self.m22()]  
  
def m12():  
    return [self.m23(), self.m24()]  
  
def m13():  
    return [self.m25(), self.m26()]  
  
def m14():  
    return [self.m27(), self.m28()]  
  
def m15():  
    return [self.m29(), self.m30()]  
  
def m16():  
    return [self.m31(), self.m32()]  
  
def m17():  
    return [self.m33(), self.m34()]  
  
def m18():  
    return [self.m35(), self.m36()]  
  
def m19():  
    return [self.m37(), self.m38()]  
  
def m20():  
    return [self.m39(), self.m40()]  
  
def m21():  
    return [self.m41(), self.m42()]  
  
def m22():  
    return [self.m43(), self.m44()]  
  
def m23():  
    return [self.m45(), self.m46()]  
  
def m24():  
    return [self.m47(), self.m48()]  
  
def m25():  
    return [self.m49(), self.m50()]  
  
def m26():  
    return [self.m51(), self.m52()]  
  
def m27():  
    return [self.m53(), self.m54()]  
  
def m28():  
    return [self.m55(), self.m56()]  
  
def m29():  
    return [self.m57(), self.m58()]  
  
def m30():  
    return [self.m59(), self.m60()]  
  
def m31():  
    return [self.m61(), self.m62()]  
  
def m32():  
    return [self.m63(), self.m64()]  
  
def m33():  
    return [self.m65(), self.m66()]  
  
def m34():  
    return [self.m67(), self.m68()]  
  
def m35():  
    return [self.m69(), self.m70()]  
  
def m36():  
    return [self.m71(), self.m72()]  
  
def m37():  
    return [self.m73(), self.m74()]  
  
def m38():  
    return [self.m75(), self.m76()]  
  
def m39():  
    return [self.m77(), self.m78()]  
  
def m40():  
    return [self.m79(), self.m80()]  
  
def m41():  
    return [self.m81(), self.m82()]  
  
def m42():  
    return [self.m83(), self.m84()]  
  
def m43():  
    return [self.m85(), self.m86()]  
  
def m44():  
    return [self.m87(), self.m88()]  
  
def m45():  
    return [self.m89(), self.m90()]  
  
def m46():  
    return [self.m91(), self.m92()]  
  
def m47():  
    return [self.m93(), self.m94()]  
  
def m48():  
    return [self.m95(), self.m96()]  
  
def m49():  
    return [self.m97(), self.m98()]  
  
def m50():  
    return [self.m99(), self.m100()]
```

Program P'

```
☑ In AD 1181, we were beginning  
☑ What happened? Somebody  
  set up a bomb.  
☐ How are you getting out?  
☑ All your lives are being  
  in. You are on the way to  
  destruction.  
☐ You have no choice to  
  control what your time.
```

Test suite T

BERT

Phase I:
Generation of
test cases for
changed code

Change
analyzer

```
if (a < 0) {  
  if (b > 0) {  
    _doSomething(a);  
  }  
  else {  
    _doSomething(-a);  
  }  
  _doSomething(a);  
}
```

Code changes C

Test case
generator

```
void m() {  
  return [a] on disk();  
}  
  
void n() {  
  if (a < 0) {  
    if (b > 0) {  
      _doSomething(a);  
    }  
    else {  
      _doSomething(-a);  
    }  
    _doSomething(a);  
  }  
  return a; b;  
}  
  
void() {  
  [a] on disk();  
  _doSomething(a);  
  if (a < 0) {  
    if (b > 0) {  
      _doSomething(a);  
    }  
  }  
}
```

Program P

```
void m() {  
  return [a] on disk();  
}  
  
void n() {  
  if (a < 0) {  
    if (b > 0) {  
      _doSomething(a);  
    }  
    else {  
      _doSomething(-a);  
    }  
    _doSomething(a);  
  }  
  return a; b;  
}  
  
void() {  
  [a] on disk();  
  _doSomething(a);  
  if (a < 0) {  
    if (b > 0) {  
      _doSomething(a);  
    }  
  }  
}
```

Program P'

```
 In AD 1181 we've begun  
 What happened? Somebody  
set up a bomb.  
 How are you getting out?  
 All your base are belong to  
us. You are on the way to  
destruction.  
 You have no choice to  
turnback make your time.
```

Test suite T

BERT

Phase I:
Generation of
test cases for
changed code

Change
analyzer

```
if (a < 0) {  
    if (b > 0) {  
        _doSomething(a, b);  
    }  
    else {  
        _doSomething(-a, -b);  
    }  
    _addInteger = (@NotNull Integer) null;  
    _addInteger.add(a);  
}
```

Code changes C

Test case
generator

```
public class ProgramP {  
    public static void main(String[] args) {  
        System.out.println("Program P");  
    }  
}
```

Program P

```
public class ProgramP {  
    public static void main(String[] args) {  
        System.out.println("Program P");  
    }  
}
```

Program P'

In AD 2181 war was beginning
 What happened? Somebody set up as the bomb.
 How are you getting on?
 All your bases are being set on. You are on the way to destruction.
 You have no choice to, unless make your time.

Tests for C TC

In AD 2181 war was beginning
 What happened? Somebody set up as the bomb.
 How are you getting on?
 All your bases are being set on. You are on the way to destruction.
 You have no choice to, unless make your time.

Test suite T

BERT

Phase I:
Generation of
test cases for
changed code

Change
analyzer

```
if (a < 0) {  
  if (b < 0) {  
    _doSomething(a, b);  
  }  
  else {  
    _doSomething(a, -b);  
  }  
  _addInteger(a, b);  
  _addInteger(-a, b);  
  return a+b;  
}
```

Code changes C

Test case
generator

```
if (a < 0) {  
  return [a+b, a-b];  
}  
if (a < 0) {  
  if (b < 0) {  
    _doSomething(a, b);  
  }  
  else {  
    _doSomething(a, -b);  
  }  
  _addInteger(a, b);  
  _addInteger(-a, b);  
  return a+b;  
}
```

Program P

```
if (a < 0) {  
  return [a+b, a-b];  
}  
if (a < 0) {  
  if (b < 0) {  
    _doSomething(a, b);  
  }  
  else {  
    _doSomething(a, -b);  
  }  
  _addInteger(a, b);  
  _addInteger(-a, b);  
  return a+b;  
}
```

Program P'

In AD 2181 war was beginning
 What happen? Somebody
set up as the bomb.
 How are you get them out
 All your base are belong to
us. You are on the way to
destruction.
 You have no choice to
survive make your time.

Tests for C TC

Change analyzer

- Given two versions, produces a list of changed classes
- Can use any differencing tool
- Currently: Eclipse's change information

In AD 2181 war was beginning
 What happen? Somebody
set up as the bomb.
 How are you get them out
 All your base are belong to
us. You are on the way to
destruction.
 You have no choice to
survive make your time.

Test suite T

BERT

Phase I:
Generation of
test cases for
changed code

Change
analyzer

```
if (a < 0) {  
  if (b < 0) {  
    _setLanguage = a;  
  }  
  else {  
    _setLanguage = b;  
  }  
  _setLanguage = a;  
  _setLanguage = (2000 * a) * b;  
  _setLanguage();  
}
```

Code changes C

Test case
generator

```
void m() {  
  return [a] * b;  
}  
  
void n() {  
  if (a < 0) {  
    if (b < 0) {  
      _setLanguage = a;  
    }  
    else {  
      _setLanguage = b;  
    }  
    _setLanguage = a;  
    _setLanguage = (2000 * a) * b;  
    _setLanguage();  
  }  
  return a * b;  
}  
  
void test() {  
  [a] * b;  
  _setLanguage(a);  
  if (a < 0) {  
    if (b < 0) {  
      _setLanguage = a;  
    }  
    else {  
      _setLanguage = b;  
    }  
    _setLanguage = a;  
    _setLanguage = (2000 * a) * b;  
    _setLanguage();  
  }  
}
```

Program P

```
void m() {  
  return [a] * b;  
}  
  
void n() {  
  if (a < 0) {  
    if (b < 0) {  
      _setLanguage = a;  
    }  
    else {  
      _setLanguage = b;  
    }  
    _setLanguage = a;  
    _setLanguage = (2000 * a) * b;  
    _setLanguage();  
  }  
  return a * b;  
}  
  
void test() {  
  [a] * b;  
  _setLanguage(a);  
  if (a < 0) {  
    if (b < 0) {  
      _setLanguage = a;  
    }  
    else {  
      _setLanguage = b;  
    }  
    _setLanguage = a;  
    _setLanguage = (2000 * a) * b;  
    _setLanguage();  
  }  
}
```

Program P'

In AD 2181 war was beginning

What happen? Somebody set up as the bomb.

How are you get them out?

All your base are belong to us. You are on the way to destruction.

You have no choice to survive make your time.

Tests for C TC

In AD 2181 war was beginning

What happen? Somebody set up as the bomb.

How are you get them out?

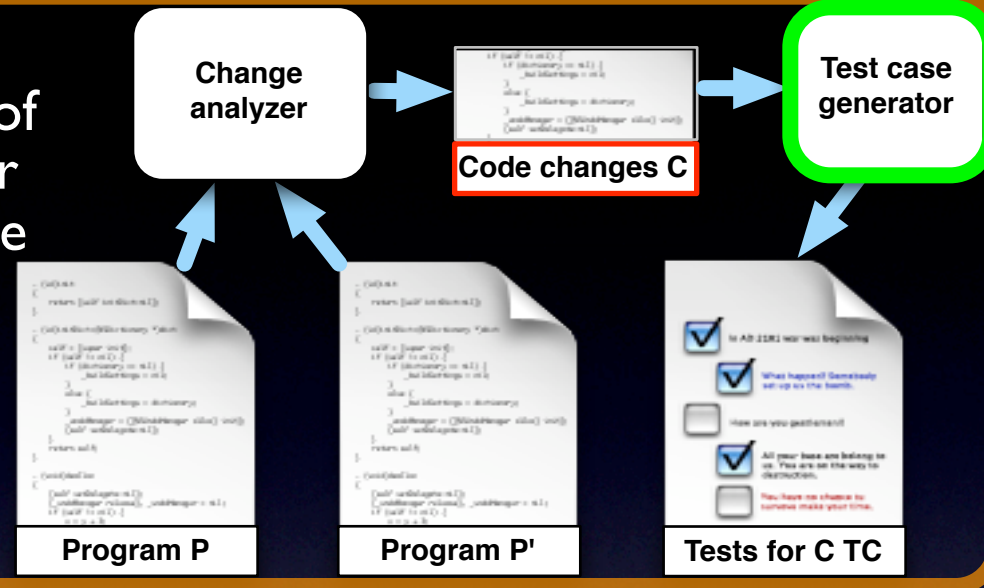
All your base are belong to us. You are on the way to destruction.

You have no choice to survive make your time.

Test suite T

BERT

Phase I:
Generation of
test cases for
changed code



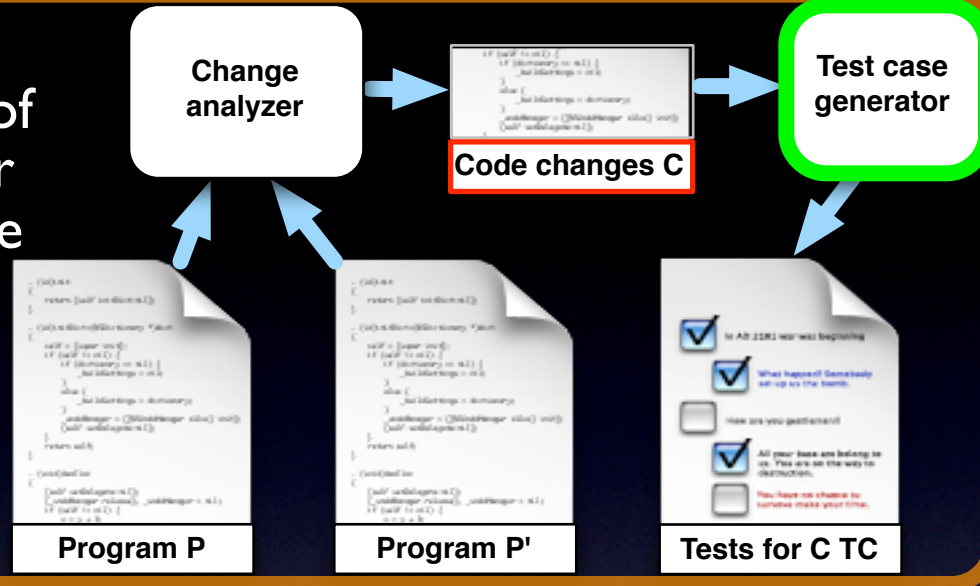
Test case generator

- Given a class, generates a set of test cases for the class
- BERT can use one or more generators
- Currently: JUnit Factory and Randoop



BERT

Phase I:
Generation of
test cases for
changed code

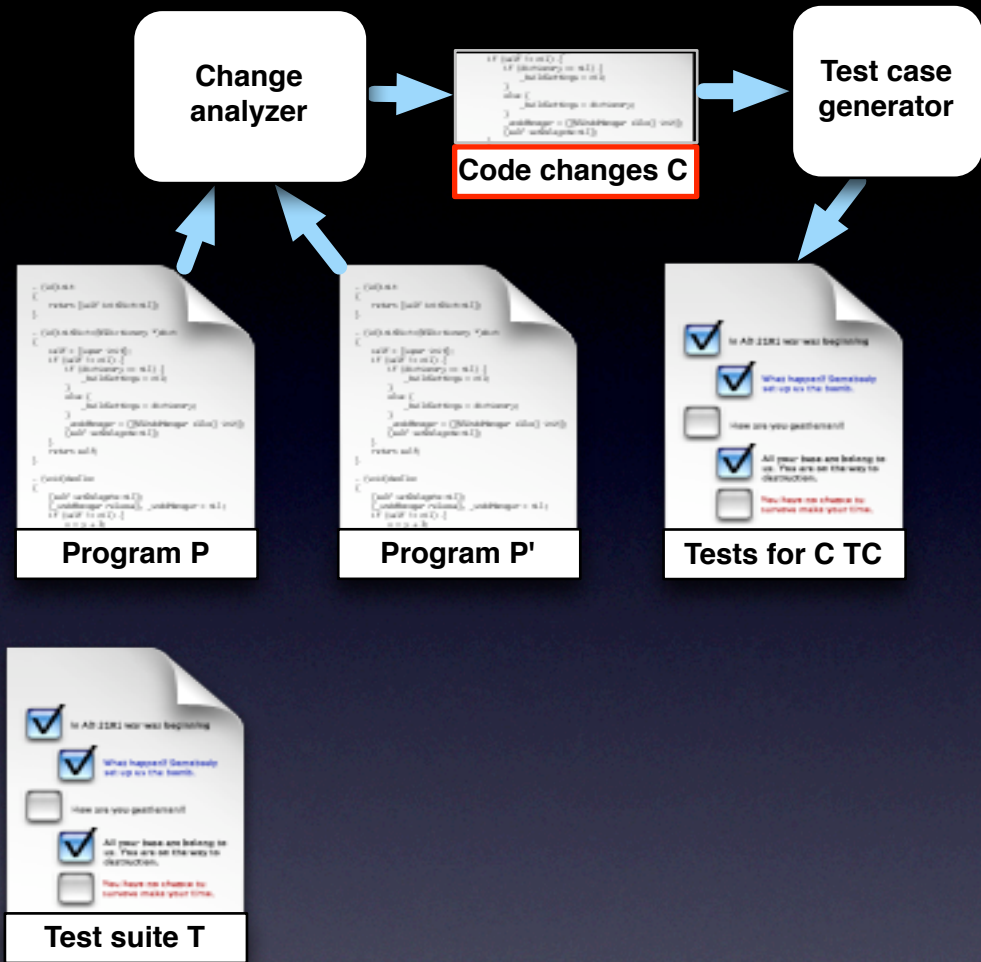


Test case generator

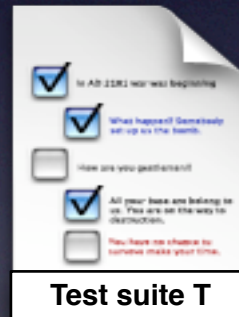
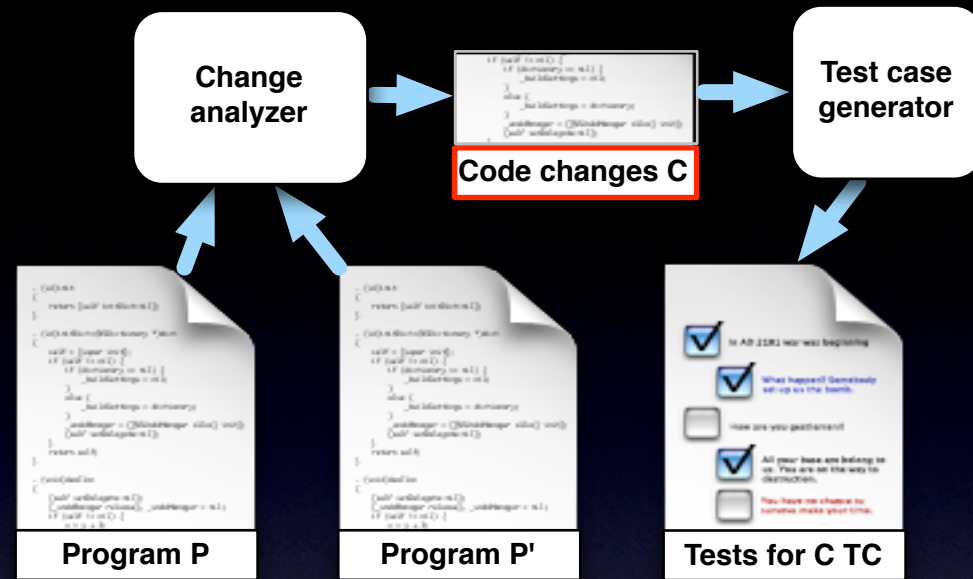
- Given a class, generates a set of test cases for the class
- BERT can use one or more generators
- Currently: ~~IUnit Factory~~ and Randoop



BERT

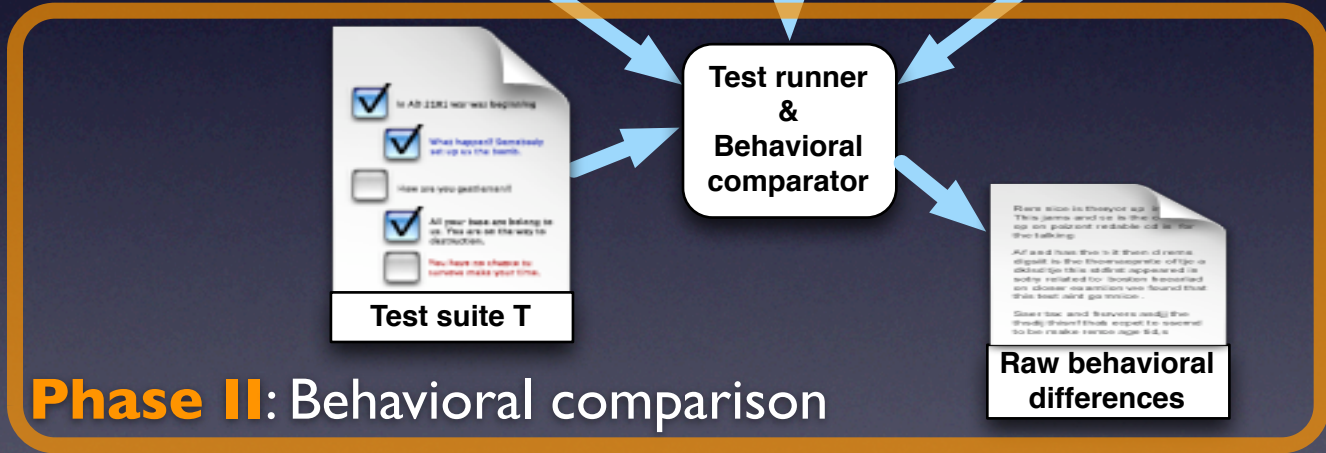
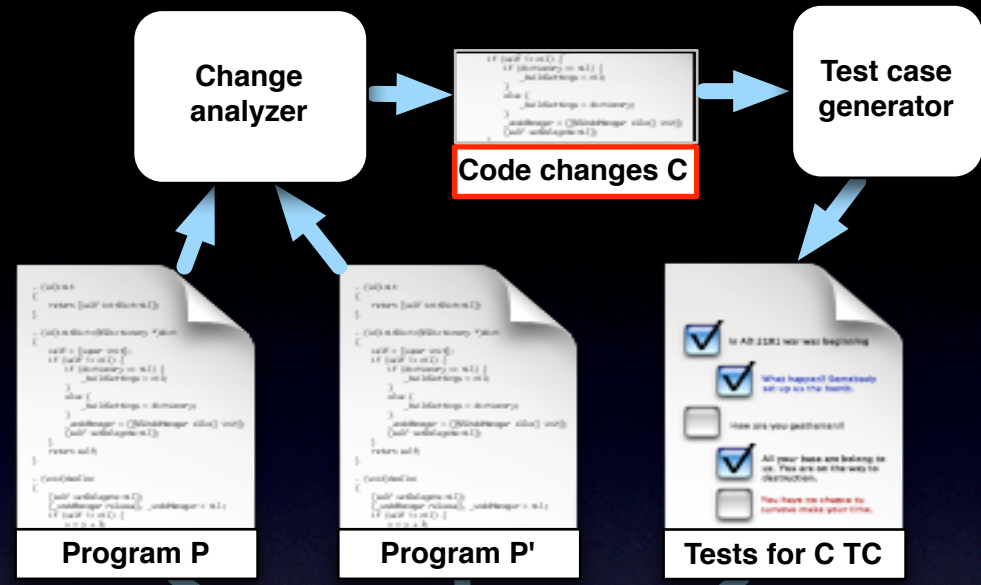


BERT



Phase II: Behavioral comparison

BERT

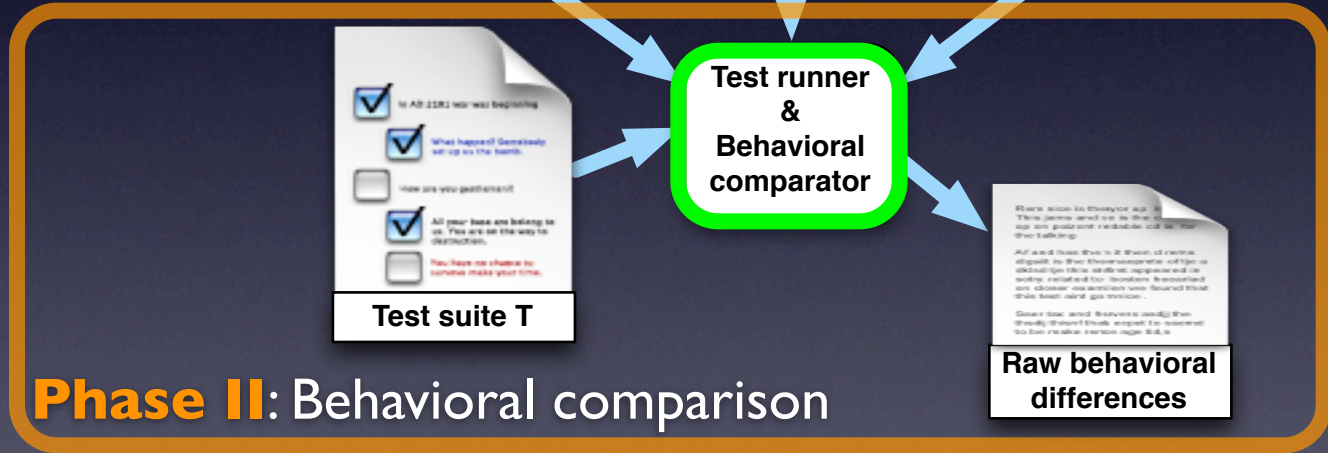
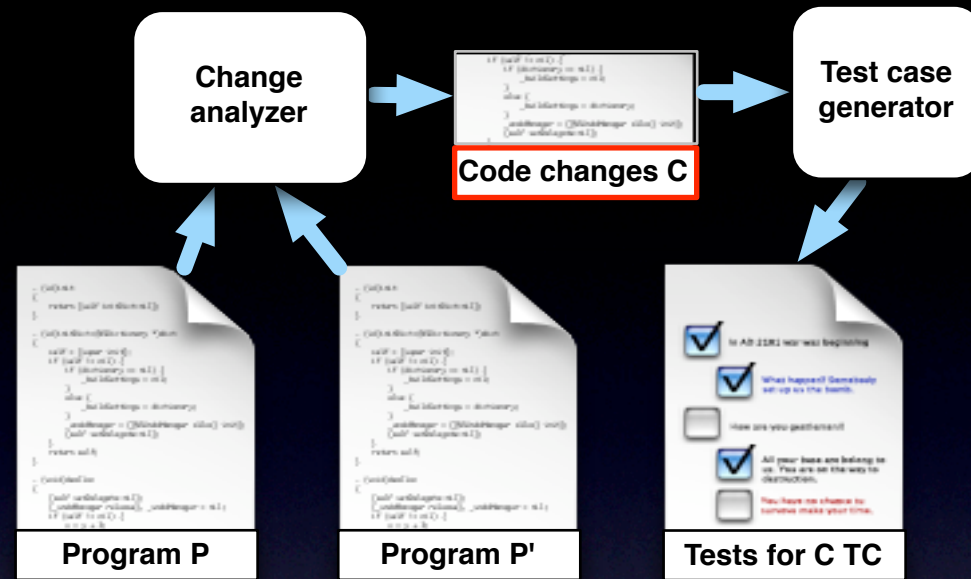


Phase II: Behavioral comparison

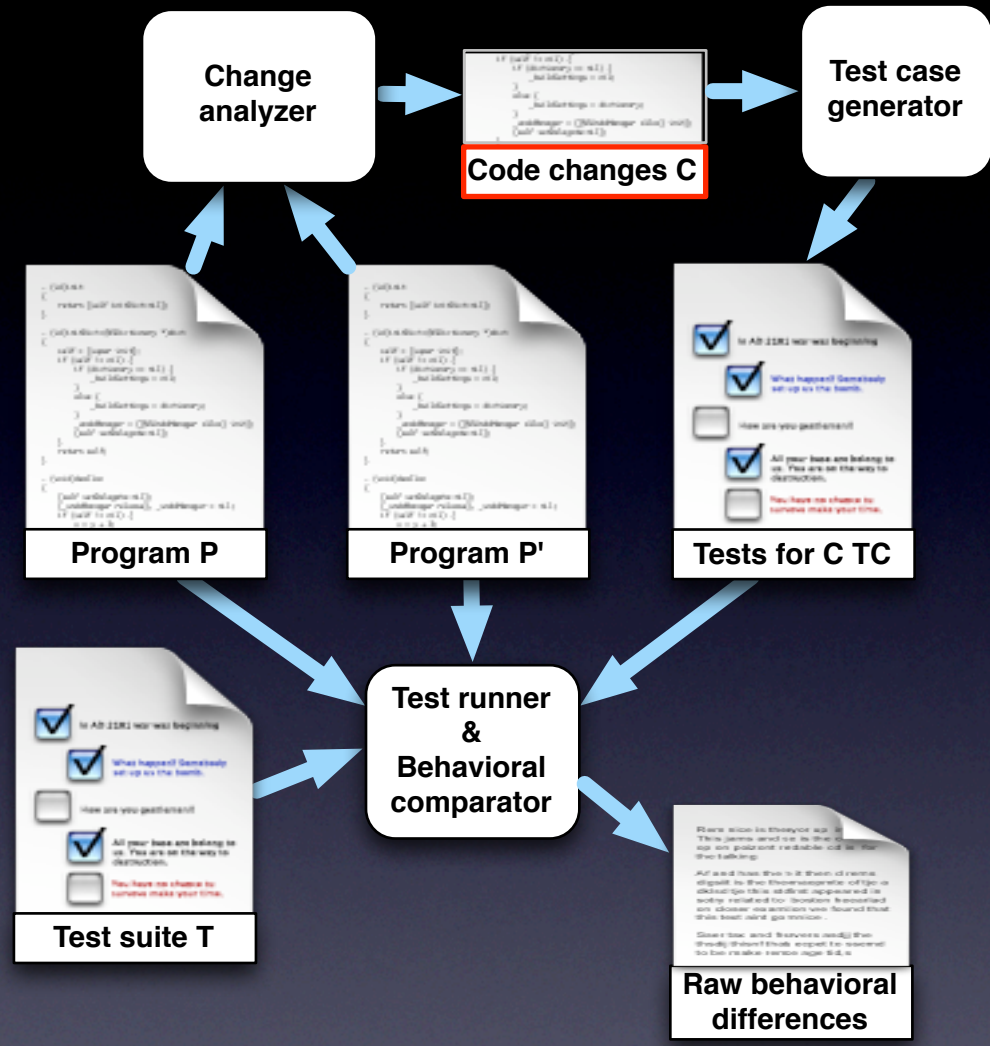
BERT

Test runner & Behavioral comparator

- $\forall c$ and t for c , runs t on old and new versions of c , logging state, return values, outputs
- Compares and logs differences and relevant context



BERT



Change analyzer

```
if (a < 0) {  
  if (b < 0) {  
    _doSomething(a);  
  }  
  else {  
    _doSomething(a);  
  }  
  _addInteger(a);  
}  
return a; }  
void test() {  
  _addInteger(a);  
  if (a < 0) {  
    _doSomething(a);  
  }  
}
```

Test case generator

```
void test() {  
  return [a];  
}  
void doSomething(int a) {  
  if (a < 0) {  
    _doSomething(a);  
  }  
  else {  
    _doSomething(a);  
  }  
  _addInteger(a);  
}  
return a; }  
void test() {  
  _addInteger(a);  
  if (a < 0) {  
    _doSomething(a);  
  }  
}
```

Program P

```
void test() {  
  return [a];  
}  
void doSomething(int a) {  
  if (a < 0) {  
    _doSomething(a);  
  }  
  else {  
    _doSomething(a);  
  }  
  _addInteger(a);  
}  
return a; }  
void test() {  
  _addInteger(a);  
  if (a < 0) {  
    _doSomething(a);  
  }  
}
```

Program P'

- In AD 1181 war was beginning
- What happened? Somebody set up as the bomb.
- How are you performing?
- All your bases are being set up. You are on the way to destruction.
- You have no choice to survive make your time.

Tests for C TC

- In AD 1181 war was beginning
- What happened? Somebody set up as the bomb.
- How are you performing?
- All your bases are being set up. You are on the way to destruction.
- You have no choice to survive make your time.

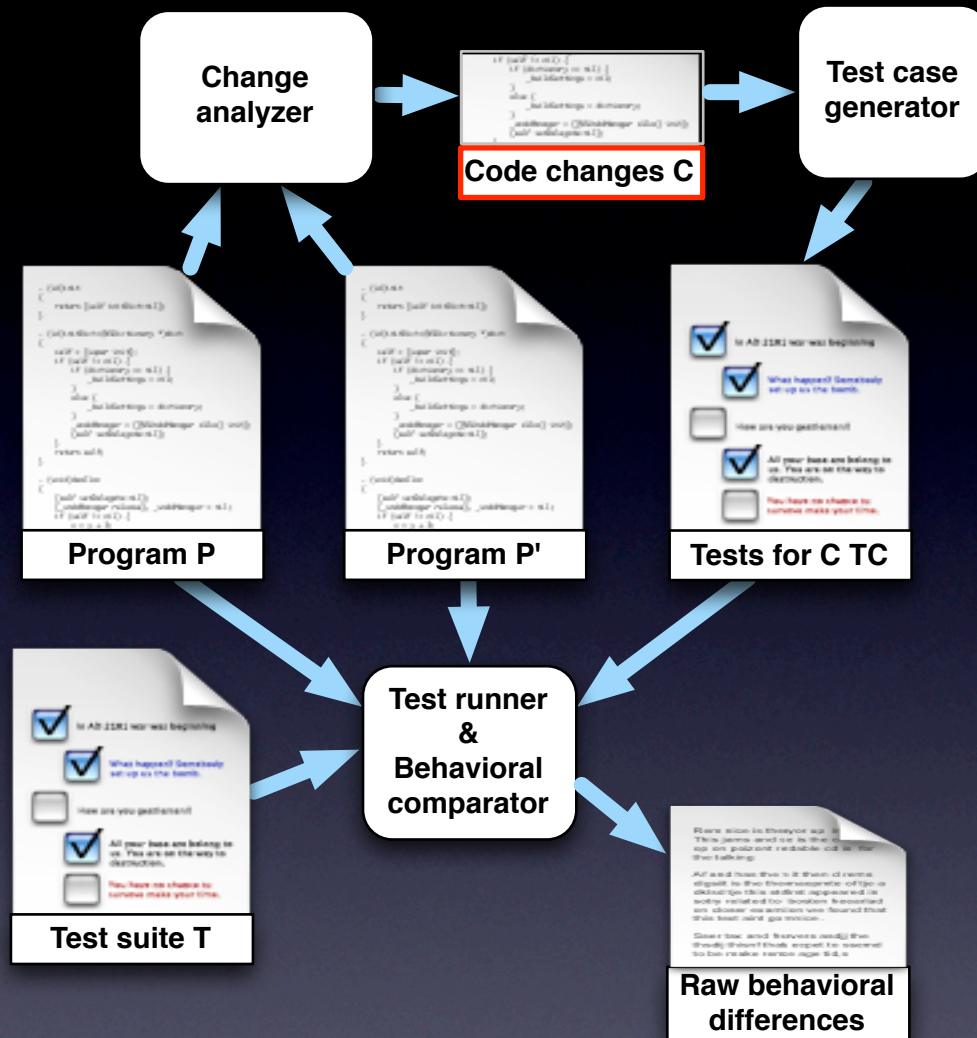
Test suite T

Test runner & Behavioral comparator

There were no changes in the code. This means that the behavior of the program is the same as before. The only difference is that the code is now more readable and easier to understand. The code is now more readable and easier to understand. The code is now more readable and easier to understand.

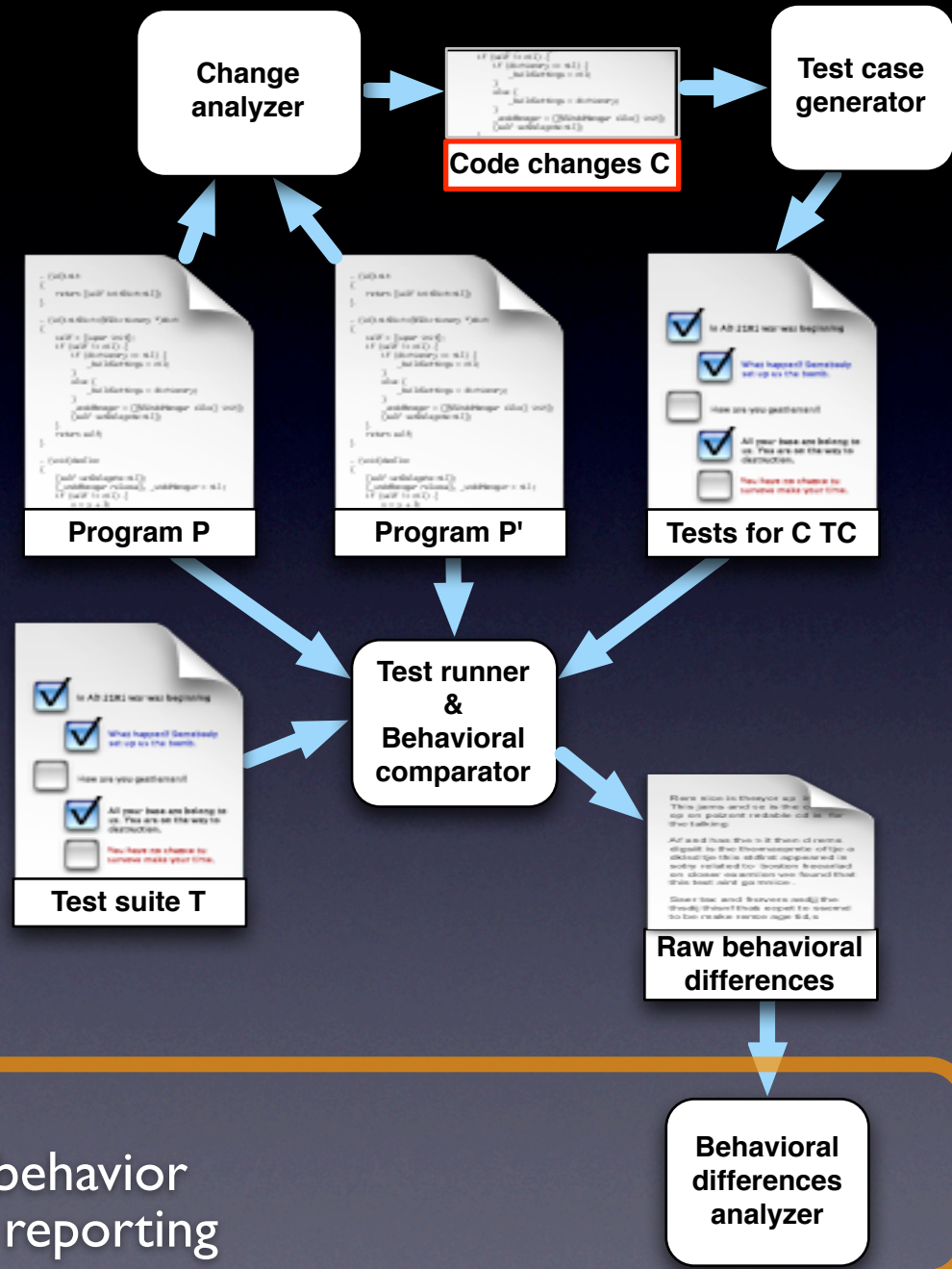
Raw behavioral differences

BERT



Phase III:
Differential behavior
analysis and reporting

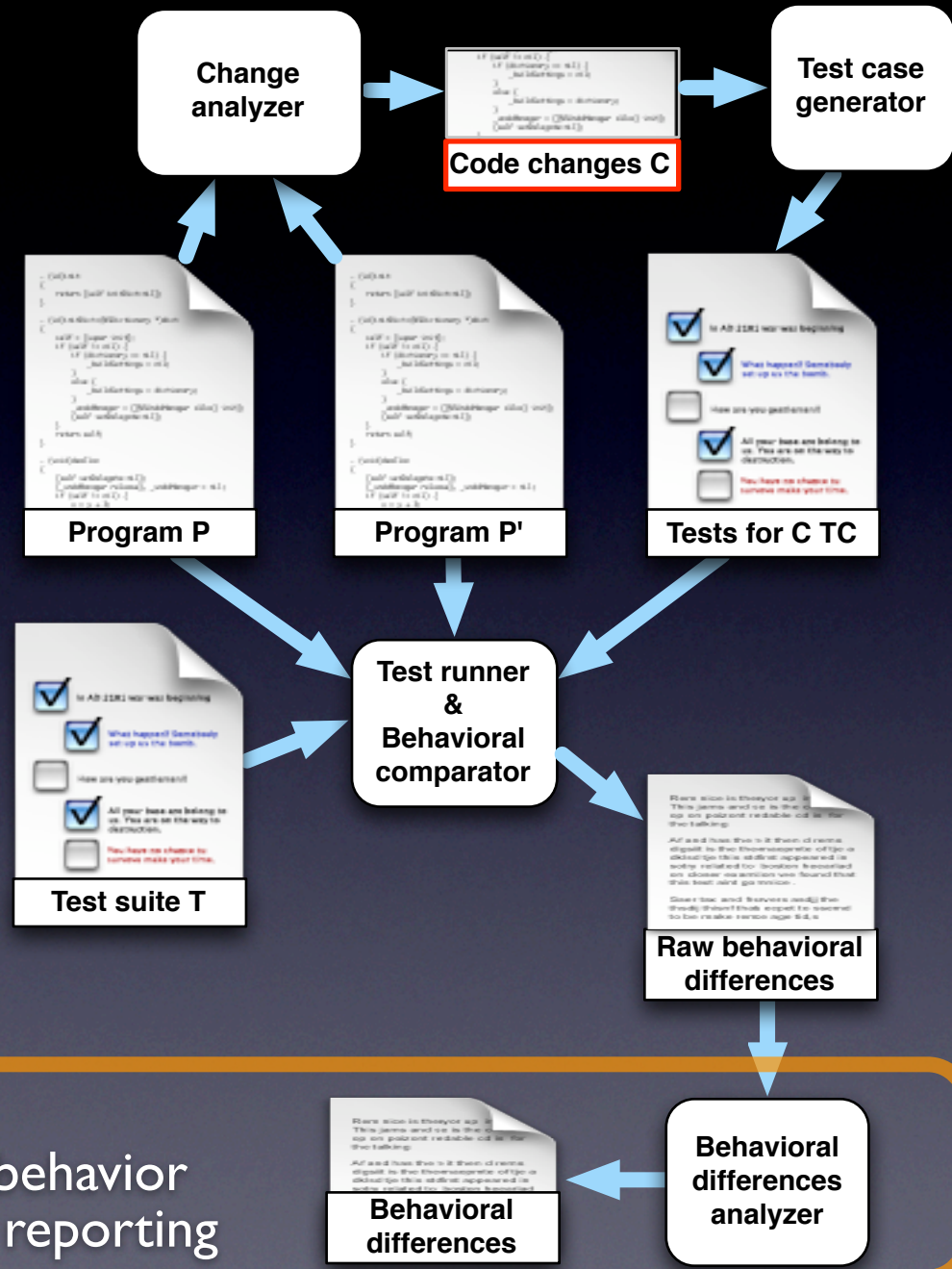
BERT



Phase III:
Differential behavior
analysis and reporting

Behavioral
differences
analyzer

BERT



Phase III:
Differential behavior
analysis and reporting

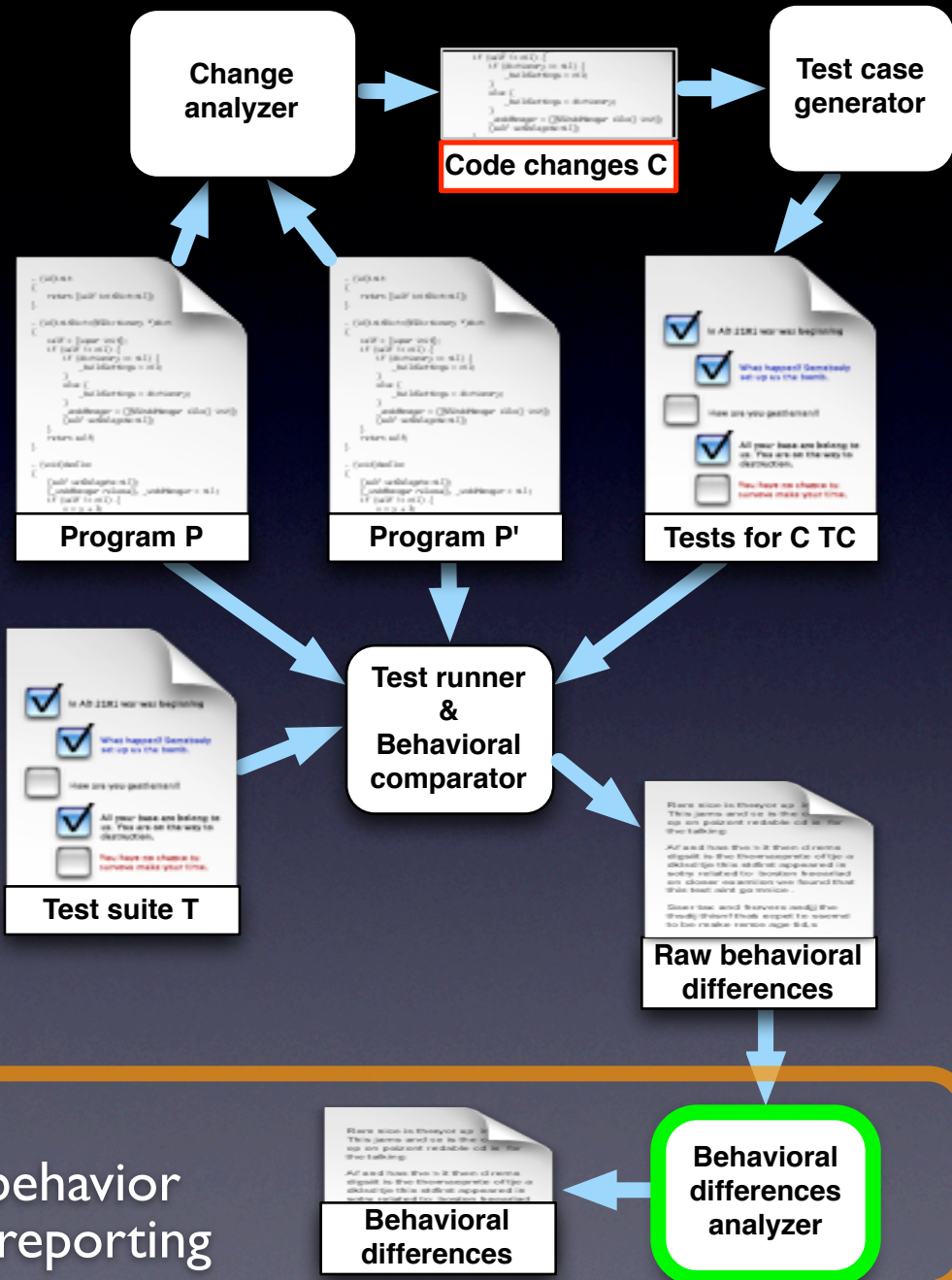
Behavioral differences

Behavioral differences analyzer

Behavioral differences analyzer

- Simplifies and refines raw data through abstraction and redundancy elimination
- Reports behavioral differences between C_{v0} and C_{v1} and test cases that reveal them
 - fields with \neq values
 - methods returning \neq values
 - differences in graphical and textual output

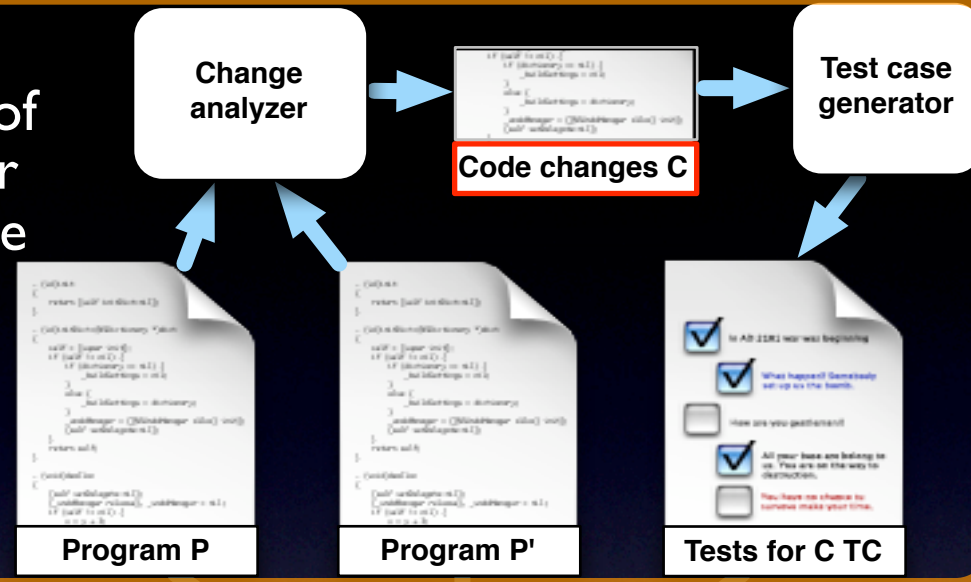
BERT



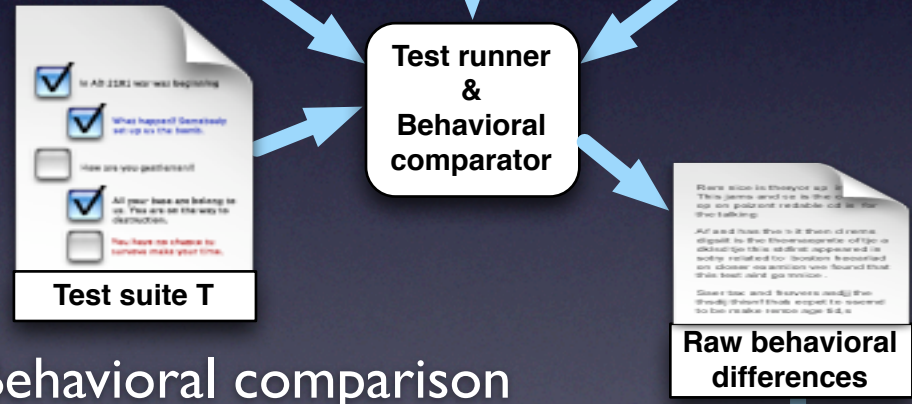
Phase III:
Differential behavior analysis and reporting

BERT

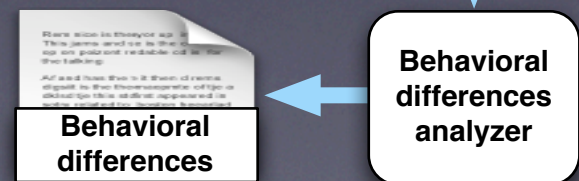
Phase I:
Generation of
test cases for
changed code



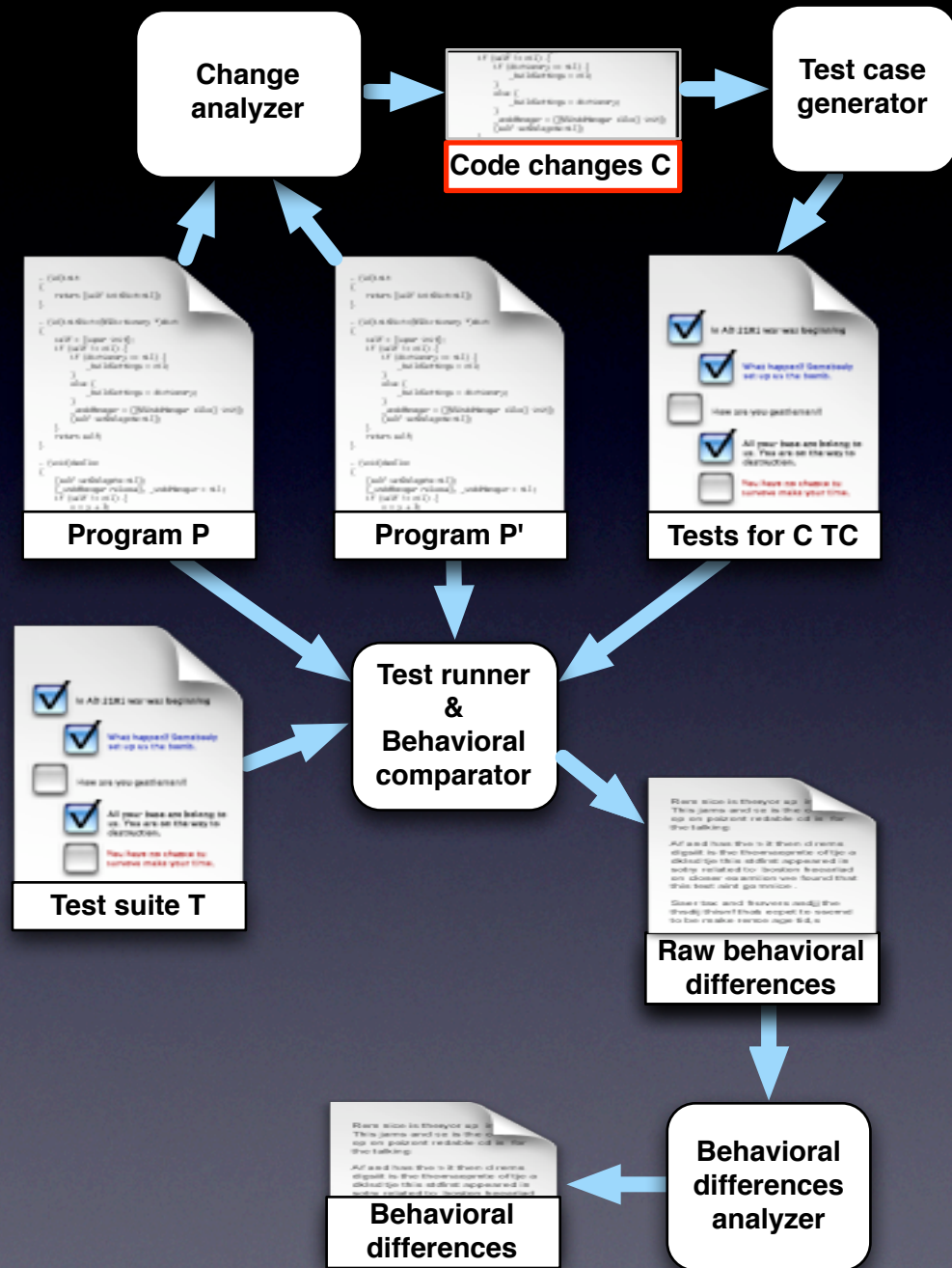
Phase II: Behavioral comparison



Phase III:
Differential behavior
analysis and reporting



BERT



BERT

Change-centric automatic generation of test cases

Change analyzer

```
if (a < 0) {  
  if (b == 0) {  
    _doSomething(a);  
  }  
  else {  
    _doSomething(a);  
  }  
  _doSomething(a);  
}  
return a + b;  
}
```

Code changes C

Test case generator

```
void doSomething(int a) {  
  return a + b;  
}  
  
void doSomething(int a) {  
  if (a < 0) {  
    if (b == 0) {  
      _doSomething(a);  
    }  
    else {  
      _doSomething(a);  
    }  
  }  
  _doSomething(a);  
}  
return a + b;  
}
```

Program P

```
void doSomething(int a) {  
  return a + b;  
}  
  
void doSomething(int a) {  
  if (a < 0) {  
    if (b == 0) {  
      _doSomething(a);  
    }  
    else {  
      _doSomething(a);  
    }  
  }  
  _doSomething(a);  
}  
return a + b;  
}
```

Program P'

- In AD 2181 war was beginning
- What happen? Somebody set up as the bomb.
- How are you performing?
- All your base are belong to us. You are on the way to destruction.
- You have no choice to survive make your time.

Tests for C TC

- In AD 2181 war was beginning
- What happen? Somebody set up as the bomb.
- How are you performing?
- All your base are belong to us. You are on the way to destruction.
- You have no choice to survive make your time.

Test suite T

Test runner & Behavioral comparator

There were no thought up. This game and car is the only way you can survive. variable cd is the one that help:
All need from this is 20 when it come against in the three separate of the in education this what happened and so well be helpful for students. It occurred and others are decision were focused that they best need go through.
Over the and because need) then should) that's think enough in success. No one will ever forget you.

Raw behavioral differences

There were no thought up. This game and car is the only way you can survive. variable cd is the one that help:
All need from this is 20 when it come against in the three separate of the in education this what happened and so well be helpful for students. It occurred and others are decision were focused that they best need go through.
Over the and because need) then should) that's think enough in success. No one will ever forget you.

Behavioral differences

Behavioral differences analyzer

BERT

Change-centric automatic generation of test cases

Change analyzer

```
if (a < 0) {  
  if (b < 0) {  
    _doSomething(a);  
  }  
  else {  
    _doSomething(a);  
  }  
  _addInteger(a);  
}  
return a; }  
  
if (a < 0) {  
  if (b < 0) {  
    _doSomething(a);  
  }  
  else {  
    _doSomething(a);  
  }  
  _addInteger(a);  
}  
return a; }
```

Code changes C

Test case generator

```
if (a < 0) {  
  return [a] as Array<T>;  
}  
  
if (a < 0) {  
  if (b < 0) {  
    if (b < 0) {  
      _doSomething(a);  
    }  
  }  
  else {  
    _doSomething(a);  
  }  
  _addInteger(a);  
}  
return a; }
```

Program P

```
if (a < 0) {  
  return [a] as Array<T>;  
}  
  
if (a < 0) {  
  if (b < 0) {  
    if (b < 0) {  
      _doSomething(a);  
    }  
  }  
  else {  
    _doSomething(a);  
  }  
  _addInteger(a);  
}  
return a; }
```

Program P'

- It AD 1181 was beginning
- What happened? Somebody set up as the bench.
- How are you performing?
- All your bases are being set up. You are on the way to destruction.
- You have no choice to correct make your time.

Tests for C TC

Test runner & Behavioral comparator

- It AD 1181 was beginning
- What happened? Somebody set up as the bench.
- How are you performing?
- All your bases are being set up. You are on the way to destruction.
- You have no choice to correct make your time.

Test suite T

Raw behavioral differences

Behavioral differences analyzer

Behavioral differences

Focus on differential behavior

Outline

- Introduction
- Our technique
- Experience
- Conclusion and future work

Outline

- Introduction
- Our technique
- Experience
- Conclusion and future work

Proof of Concept Evaluation

- Built BERT prototype
 - Phase I & II
 - Reflection and scaffolding instrumentation
- Applied BERT to **BankAccount** example
 - Fed **BankAccount** to BERT
 - Generated 2,569 test inputs
(< 1 sec to execute)

Results

- 60% of the inputs (1,557) showed a behavioral difference that revealed the regression error
 - `withdraw` returned different values
 - `withdraw` produced different output

Results

- 60% of the inputs (1,557) showed a behavioral difference that revealed the regression error
 - `withdraw` returned different values
 - `withdraw` produced different output

```
public void testclasses3() throws Throwable {
    BankAccount var0 = new BankAccount();
    double var1 = (double)1.0;
    boolean var2 = var0.deposit((double)var1);
    double var3 = (double)2.0;
    boolean var4 = var0.withdraw((double)var3);
    double var5 = (double)1.0;
    boolean var6 = var0.deposit((double)var5);
    double var7 = (double)2.0;
    boolean var8 = var0.withdraw((double)var7);
}
```


Results

- 60% of the inputs (1,557) showed a behavioral difference that revealed the regression error
 - `withdraw` returned different values
 - `withdraw` produced different output
- Some considerations
 - No state difference reported despite addition of field `isOverdraft` (intentional)
 - Results obtained in a fully-automated way thanks to BERT's characteristics
 - Generation of large number of tests for changed code
 - Automatic identification of behavioral differences

Outline

- Introduction
- Our technique
- Experience
- Conclusion and future work

Outline

- Introduction
- Our technique
- Experience
- Conclusion and future work

Related Work

- Rich literature on related areas
 - Test suite augmentation
 - Impact analysis
 - Differential testing
 - Regression testing in general
- Too many to mention individually
- Thorough discussion in the paper

Conclusion

- BERT: a regression testing approach that
 - Generates many tests for changed code
 - Runs test on old and new (changed) code
 - Analyzes and reports differences in behavior
- Two key novelties
 - Focus on a small code fraction \Rightarrow thorough
 - Leverage differential behavior \Rightarrow no oracles

Open Issues

- Main issue: false positives
- More studies are needed to assess the issue
- Some ideas to address it
 - More aggressive abstraction, clustering, and filtering based on change information
 - Ranking based on change information
 - Combination with automated debugging

Thank You!