Feature Consistency in Compile-Time–Configurable System Software
Facing the Linux 10000 Feature Problem

Reinhard Tartler, Daniel Lohmann, Julio Sincero, Wolfgang Schröder-Preikschat

System Software Group
Friedrich-Alexander University
Erlangen-Nuremberg

April 11, 2011
System Software is incredibly configurable
System Software is Configurable

- System Software is incredibly configurable
- Complexity increases considerably
System Software is incredibly configurable

Complexity increases considerably

Source of **bugs**!
System Software is Configurable

Linux v2.6.35 contains:

11.057 Features
27.166 Source files
82.116 #ifdef blocks
Variability Implementation in Linux

```
#ifdef CONFIG_HOTPLUG_CPU
...
#endif
```

source files
KConfig files
config HOTPLUG_CPU
  bool "Support for ...
  depends on SMP && ...

user selection

```
#define CONFIG_HOTPLUG_CPU
#define CONFIG_SMP
...```

cmake

```
Kconfig
```

implementation variant
implementation space

```
autoconf.h
```

gcc

```
Kbuild
```

configuration variant

configuration space

```
kconfig
```

Variability Implementation in Linux

1. KConfig files
2. User selection
3. Kconfig
4. Kbuild

configuration space

configuration variant

KConfig files

source files

implementation space

implementation variant

user selection

auto.make

#define CONFIG_HOTPLUG_CPU
#define CONFIG_SMP

gcc
Variability Implementation in Linux

configuration space

1. `config HOTPLUG_CPU`  
   `bool "Support for ..."`  
   `depends on SMP && ...`

KConfig files

Kconfig

user selection

2. `auto.make`

Kbuild

configuration variant

3. `#ifdef CONFIG_HOTPLUG_CPU`  
   `...`  
   `#endif`

source files

implementation space

4. `#define CONFIG_HOTPLUG_CPU`  
   `#define CONFIG_SMP`  
   `...`

autoconf.h

gcc

implementation variant
The Problem

Configuration

Implementation
Source of Inconsistencies!
Finding Bugs with Tools for Static Analysis

Bugs in declaration and implementation

Excellent tool support for static analysis:

- Dingo: Taming Device Drivers (EuroSys’09)
- KLEE: Automatic generation of high-coverage tests (EuroSys’08)
- RWset: Attacking path explosion (TACAS’08)
- EXE: Automatically generating inputs of death (CCS’06)
- ...

Finding Bugs with Tools for Static Analysis

- Bugs in **declaration** and **implementation**

- Excellent tool support for **static analysis**:
  - Dingo: Taming Device Drivers (EuroSys’09)
  - KLEE: Automatic generation of high-coverage tests (EuroSys’08)
  - RWset: Attacking path explosion (TACAS’08)
  - EXE: Automatically generating inputs of death (CCS’06)
  - ...

- Each of them is configuration **agnostic**.
1. Introduction

2. Analysis

3. Approach and Implementation

4. Results

5. Future Work and Conclusions
Problem Analysis

Configuration

Implementation
Problem Analysis

Configuration

Implementation

symbols

constraints

symbols

constraints
Problem Analysis

**Configuration**

```
config HOTPLUG_CPU
```

**Implementation**

```
depends on SMP && HOTPLUG
```
Problem Analysis

Configuration

- config HOTPLUG_CPU
- symbols
- constraints

depends on SMP && HOTPLUG

Implementation

- #ifdef CONFIG_CPU_HOTPLUG
- symbols
- constraints

#ifdef CONFIG_CPU_HOTPLUG
#else
#endif
Problem Analysis

Configuration

Implementation

config HOTPLUG_CPU

#ifdef CONFIG_CPU_HOTPLUG

symbols

constraints

depends on SMP & HOTPLUG

#ifdef CONFIG_CPU_HOTPLUG

symbols

constraints

#ifdef CONFIG_CPU_HOTPLUG

#endif

#else

#endif
Problem Analysis

Configuration

Implementation

Symbolic ⊤

config HOTPLUG_CPU

depends on SMP && HOTPLUG

IFDEF CONFIG_CPU_HOTPLUG

IFDEF CONFIG_CPU_HOTPLUG

ELSE

ENDIF

symbols

constraints

symbols

constraints
Problem Analysis

Configuration

Implementation

Symbolic ⊈

Logic ⊈

depends on SMP && HOTPLUG

config HOTPLUG_CPU

#ifdef CONFIG_CPU_HOTPLUG
#ifdef CONFIG_CPU_HOTPLUG
#else
#endif

#endif CONFIG_CPU_HOTPLUG

# ifdef CONFIG_CPU_HOTPLUG
# else
# endif
Symbolic Inconsistency

```c
static int hotplug_cfd (struct notifier_block *nfb, unsigned long action, void *hcpu)
{
    // ...
    switch (action) {
        case CPU_UP_PREPARE:
        case CPU_UP_PREPARE_FROZEN:
        // ...
        #ifdef CONFIG_CPU_HOTPLUG
        case CPU_UP_CANCELED:
        case CPU_UP_CANCELED_FROZEN:
        case CPU_DEAD:
        case CPU_DEAD_FROZEN:
            free_cpumask_var(cfd -> cpumask);
            break;
        #endif
    }
    return NOTIFY_OK;
}
```

Result: Fix for a critical bug
Symbolic Inconsistency

```c
static int
hotplug_cfd(struct notifier_block *nfb, unsigned long action, void *hcpu)
{
    // [...]
    switch (action) {
    case CPU_UP_PREPARE:
    case CPU_UP_PREPARE_FROZEN:
       // [...]
    #ifdef CONFIG_CPU_HOTPLUG
    case CPU_UP_CANCELED:
    case CPU_UP_CANCELED_FROZEN:
    #endif
    case CPU_DEAD:
    case CPU_DEAD_FROZEN:
        free_cpumask_var(cfd->cpumask);
        break;
    #endif
};
return NOTIFY_OK;
```
Symbolic Inconsistency

```c
static int hotplug_cfd(struct notifier_block *nfb, unsigned long action, void *hcpu)
{
    // [...]
    switch (action) {
    case CPU_UP_PREPARE:
    case CPU_UP_PREPARE_FROZEN:
        // [...]
    #ifdef CONFIG_CPU_HOTPLUG
    case CPU_UP_CANCELED:
    case CPU_UP_CANCELED_FROZEN:
        free_cpumask_var(cfd->cpumask);
    break;
    #endif
    case CPU_DEAD:
    case CPU_DEAD_FROZEN:
        break;
    #endif
    }
    return NOTIFY_OK;
}
```

Result: Fix for a critical bug
Logic Inconsistencies

```
#ifdef CONFIG_DISCONTIGMEM
    // Block1
    static ... int pfn_to_mid(...)
#endif

#ifdef CONFIG_NUMA
    // Block2
#else
    // Block3
#endif
```

Feature DISCONTIGMEM requires NUMA

Inner block is not configuration dependent anymore

Result: code cleanup
Feature DISCONTIGMEM requires NUMA

Inner block is not configuration dependent anymore
Feature DISCONTIGMEM requires NUMA

Inner block is not configuration dependent anymore
Feature DISCONTIGMEM requires NUMA

Inner block is not configuration dependent anymore

Result: code cleanup
Outline

1. Introduction

2. Analysis

3. Approach and Implementation

4. Results

5. Future Work and Conclusions
General Approach

```c
#ifdef CONFIG_DISCONTIGMEM
    // Block 1
    static ... int pfn_to_mid(...)
#endif

#ifdef CONFIG_NUMA
    // Block 2
#else
    // Block 3
#endif
#endif
```
General Approach

```
# ifdef CONFIG_DISCONTIGMEM
    // Block1
    static ... int pfn_to_mid(...)
# ifdef CONFIG_NUMA
    // Block2
# else
    // Block3
# endif
# endif
```
**General Approach**

```
#ifdef CONFIG_DISCONTIGMEM
    // Block 1
    static ... int pfnto_mid(...)
#endif

#ifdef CONFIG_NUMA
    // Block 2
    // Block 3
    # else
    // Block 3
    # endif
    #endif
```

\[ C = (\text{FLATMEM} \rightarrow \text{MEMORY\_MODEL}) \]
\[ \land (\text{DISCONTIGMEM} \rightarrow \text{MEMORY\_MODEL}) \]
\[ \land (\text{SPARSEMEM} \rightarrow \text{MEMORY\_MODEL}) \]
\[ \land (\text{NUMA} \rightarrow \text{MEMORY\_MODEL}) \]
\[ \land (\text{DISCONTIGMEM} \rightarrow \text{NUMA}) \]
General Approach

\[ C = (\text{FLATMEM} \to \text{MEMORY\_MODEL}) \]
\[ \land (\text{DISCONTIGMEM} \to \text{MEMORY\_MODEL}) \]
\[ \land (\text{SPARSEMEM} \to \text{MEMORY\_MODEL}) \]
\[ \land (\text{NUMA} \to \text{MEMORY\_MODEL}) \]
\[ \land (\text{DISCONTIGMEM} \to \text{NUMA}) \]

\[ I = (\text{Block}_1 \leftrightarrow \text{DISCONTIGMEM}) \]
\[ \land (\text{Block}_2 \leftrightarrow \text{Block}_1 \land (\text{NUMA}) \]
\[ \land (\text{Block}_3 \leftrightarrow \text{Block}_1 \land \neg \text{Block}_2) \]
General Approach

Crosscheck both formulas with a SAT solver:

$$\text{dead?} = \text{sat}(C \land I \land \text{Block}_N)$$

$$\text{undead?} = \text{sat}(C \land I \land \neg\text{Block}_N \land \text{parent}(\text{Block}_N))$$

$$\land (\text{DISCONTIGMEM} \rightarrow \text{MEMORY\_MODEL})$$

$$\land (\text{SPARSEMEM} \rightarrow \text{MEMORY\_MODEL})$$

$$\land (\text{NUMA} \rightarrow \text{MEMORY\_MODEL})$$

$$\land (\text{DISCONTIGMEM} \rightarrow \text{NUMA})$$
Implementation Challenges

Accuracy

- Conceptually no false positives
- Exact identification of variation points
Implementation Challenges

Accuracy

- Conceptually no false positives
- Exact identification of variation points

Coverage

- Extract configuration model for all 22 architectures
- Defect $\rightsquigarrow$ detected on each architecture
Implementation Challenges

Accuracy
- Conceptually no false positives
- Exact identification of variation points

Coverage
- Extract configuration model for all 22 architectures
- Defects detected on each architecture

Performance
- Easy and fast to use during incremental builds
- Possible by problem slicing
- Complete run on Linux in 15 minutes
Implementation for Linux

```
#ifdef CONFIG_HOTPLUG_CPU
...
#endif
```

```
config HOTPLUG_CPU
  bool "Support for ..."
  depends on SMP && ...
```

Linux source

KConfig files

KConfig Parser

CPP Parser

crosscheck

SAT Engine

undertaker

whitelist filter

defect reports

Implementation for Linux

- **KConfig files**
  - `config HOTPLUG_CPU`
  - `bool "Support for ..."
  - `depends on SMP && ...`

- **Linux source**
  - `#ifdef CONFIG_HOTPLUG_CPU`
  - `...`
  - `#endif`

- **KConfig Parser**
  - **CPP Parser**
  - **crosscheck**
  - **SAT Engine**
  - **undertaker**

- **Whitelist filter**
- **Defect reports**
Implementation for Linux

Linux source

```
#define CONFIG_HOTPLUG_CPU
...
#endif
```

KConfig files

```
cfg_HOTPLUG_CPU
bool "Support for ..."
depends on SMP & & ... 
```

KConfig Parser

CPP Parser

crosscheck

SAT Engine

undertaker

whitelist filter

defect reports
Implementation for Linux

config HOTPLUG_CPU
bool "Support for ..."
depends on SMP & & ...
Outline

1. Introduction
2. Analysis
3. Approach and Implementation
4. Results
5. Future Work and Conclusions
## Results

<table>
<thead>
<tr>
<th>subsystem</th>
<th>#ifdefs</th>
<th>logic</th>
<th>symbolic</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch/</td>
<td>33757</td>
<td>345</td>
<td>581</td>
<td>926</td>
</tr>
<tr>
<td>drivers/</td>
<td>32695</td>
<td>88</td>
<td>648</td>
<td>736</td>
</tr>
<tr>
<td>fs/</td>
<td>3000</td>
<td>4</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>include/</td>
<td>7241</td>
<td>6</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>kernel/</td>
<td>1412</td>
<td>7</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>mm/</td>
<td>555</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>net/</td>
<td>2731</td>
<td>1</td>
<td>49</td>
<td>50</td>
</tr>
<tr>
<td>sound/</td>
<td>3246</td>
<td>5</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>virt/</td>
<td>53</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>other subsystems</td>
<td>601</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td><strong>∑</strong></td>
<td><strong>85291</strong></td>
<td><strong>460</strong></td>
<td><strong>1316</strong></td>
<td><strong>1776</strong></td>
</tr>
</tbody>
</table>

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>fix proposed</td>
<td>150 (1)</td>
<td>214 (22)</td>
<td>364 (23)</td>
<td></td>
</tr>
<tr>
<td>confirmed defect</td>
<td>38 (1)</td>
<td>116 (20)</td>
<td>154 (21)</td>
<td></td>
</tr>
<tr>
<td>confirmed rule-violation</td>
<td>88 (0)</td>
<td>21 (2)</td>
<td>109 (2)</td>
<td></td>
</tr>
<tr>
<td>pending</td>
<td>24 (0)</td>
<td>77 (0)</td>
<td>101 (0)</td>
<td></td>
</tr>
</tbody>
</table>
Results

<table>
<thead>
<tr>
<th>subsystem</th>
<th>ifdef</th>
<th>logic</th>
<th>symbolic</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>arch/</td>
<td>33757</td>
<td>345</td>
<td>581</td>
<td>926</td>
</tr>
<tr>
<td>drivers/</td>
<td>32695</td>
<td>88</td>
<td>648</td>
<td>736</td>
</tr>
<tr>
<td>fs/</td>
<td>3000</td>
<td>4</td>
<td>13</td>
<td>17</td>
</tr>
<tr>
<td>include/</td>
<td>7241</td>
<td>6</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>kernel/</td>
<td>1412</td>
<td>7</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>mm/</td>
<td>555</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>net/</td>
<td>2731</td>
<td>1</td>
<td>49</td>
<td>50</td>
</tr>
<tr>
<td>sound/</td>
<td>3246</td>
<td>5</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>virt/</td>
<td>53</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>other subsystems</td>
<td>601</td>
<td>4</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>∑</td>
<td>85291</td>
<td>460</td>
<td>1316</td>
<td>1776</td>
</tr>
</tbody>
</table>

We have found **1776** configurability issues

Submitted **123** patches for **364** defects

**20** are confirmed **new bugs** (affecting binary code)

Cleaned up **5129** lines of cruft code
Impact on Linux

Patch Submission during the merge window of version 2.6.36:

New and Fixed Configuration Defects over Linux Releases
Impact on Linux

Patch Submission during the merge window of version 2.6.36:
Outline

1. Introduction
2. Analysis
3. Approach and Implementation
4. Results
5. Future Work and Conclusions
Data has to be seen as lower bound:

- More precise configuration space extraction is possible
- `#define` support
- Improved implementation find $> 4000$ defects
Future Work

Data has to be seen as lower bound:

- More precise configuration space extraction is possible
- #define support
- Improved implementation find > 4000 defects

Focus on pure variability defects
Future Work

- Data has to be seen as lower bound:
  - More precise configuration space extraction is possible
  - #define support
  - Improved implementation find > 4000 defects

- Focus on pure variability defects

- Integration of configuration agnostic tools for static analysis
Conclusions

- Configurability has to be seen as a significant cause of software defects in its own respect.

- **Configuration** and **implementation** need to be kept consistent.

- Our approach finds and fixes real problems!
  - Over 100 patches submitted and about 50 accepted!
  - Excellent feedback from kernel developers.
Conclusions

- Configurability has to be seen as a significant cause of software defects in its own respect

- **Configuration** and **implementation** need to be kept consistent

- Our approach finds and fixes real problems!
  - Over **100 patches** submitted and about **50** accepted!
  - Excellent feedback from kernel developers

http://vamos.informatik.uni-erlangen.de/trac/undertaker