

Automação de teste de software para sistemas embarcados.

Orientador : Antônio Augusto Fröhlich

`guto@lisha.ufsc.br`

Mestranda: Rita de Cássia Cazu Soldi

`rita@lisha.ufsc.br`

24 de junho de 2013

Agenda



- Problema e motivação
- Objetivo
- Hipótese
- Proposta
- Metodologia
- Experimentos
 - Emular hardware
 - Automação da troca de parâmetros
 - Importar configuração para a troca de parâmentros
- Cronograma

Problema e motivação



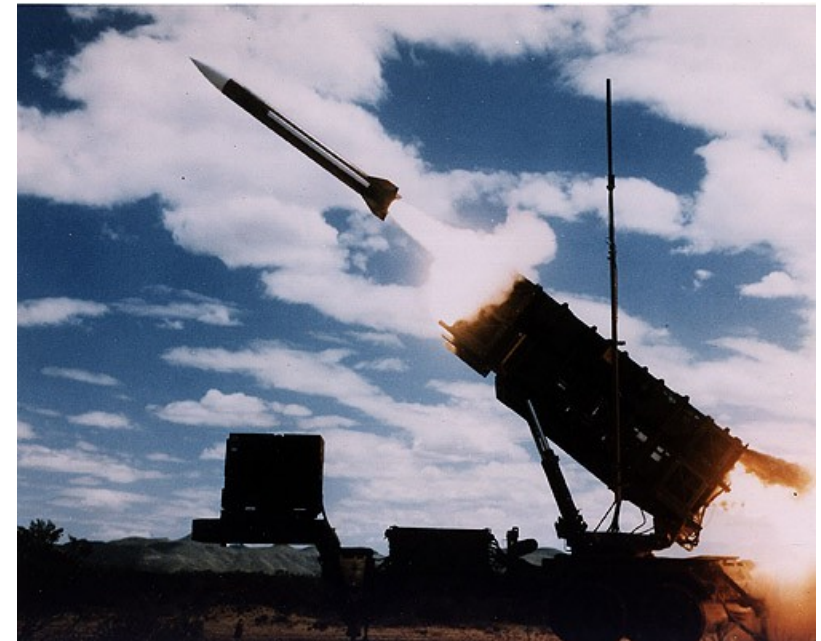
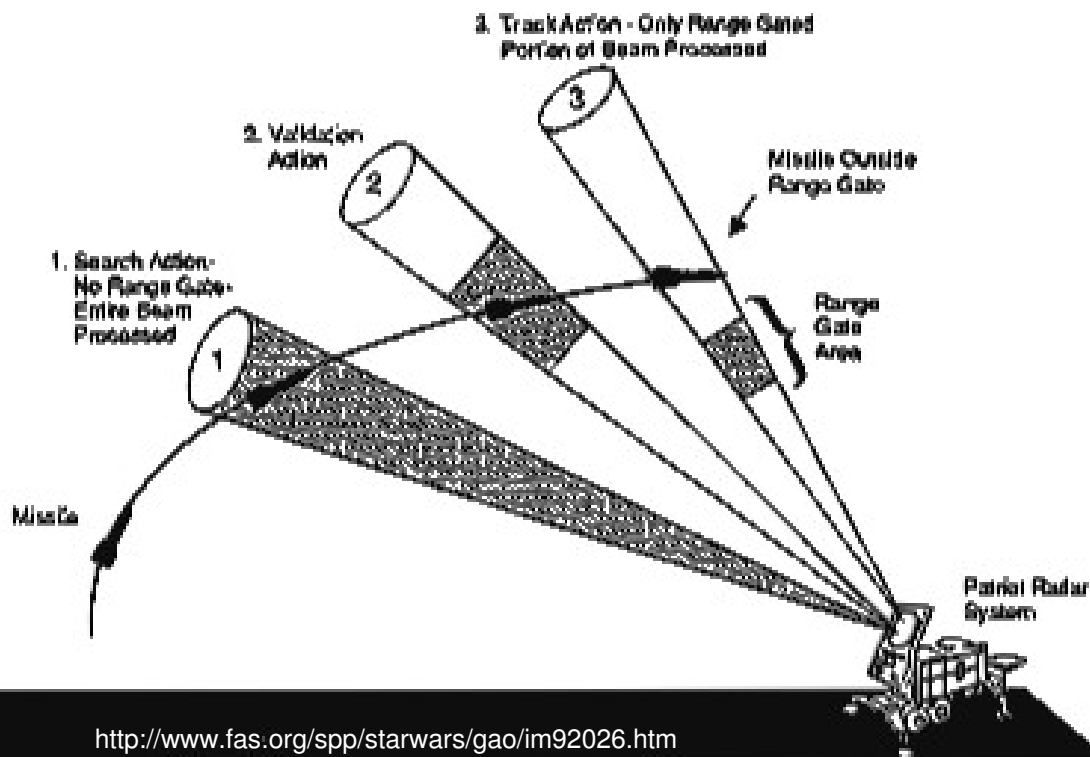
- O processo de produção de um software deve respeitar os requisitos, as especificações e os comportamentos especificados para este produto.
- Dentro deste processo existe a atividade de teste, que verifica se as especificações são atendidas.
- O objetivo dos testes é de achar erros ou falhas no software. No caso de teste reprovado, deve-se procurar a causa do erro e corrigí-lo.
 - **Problema:** atividade não-trivial e morosa.

Problema e motivação



■ Patriot (1991)

- **Causa:** Erro em arredondamento “atrasou” cálculo do relógio em 0,0034s. Depois de 100 h, a diferença era de 687 m.
- **Custo:** 28 mortos, 100 feridos.

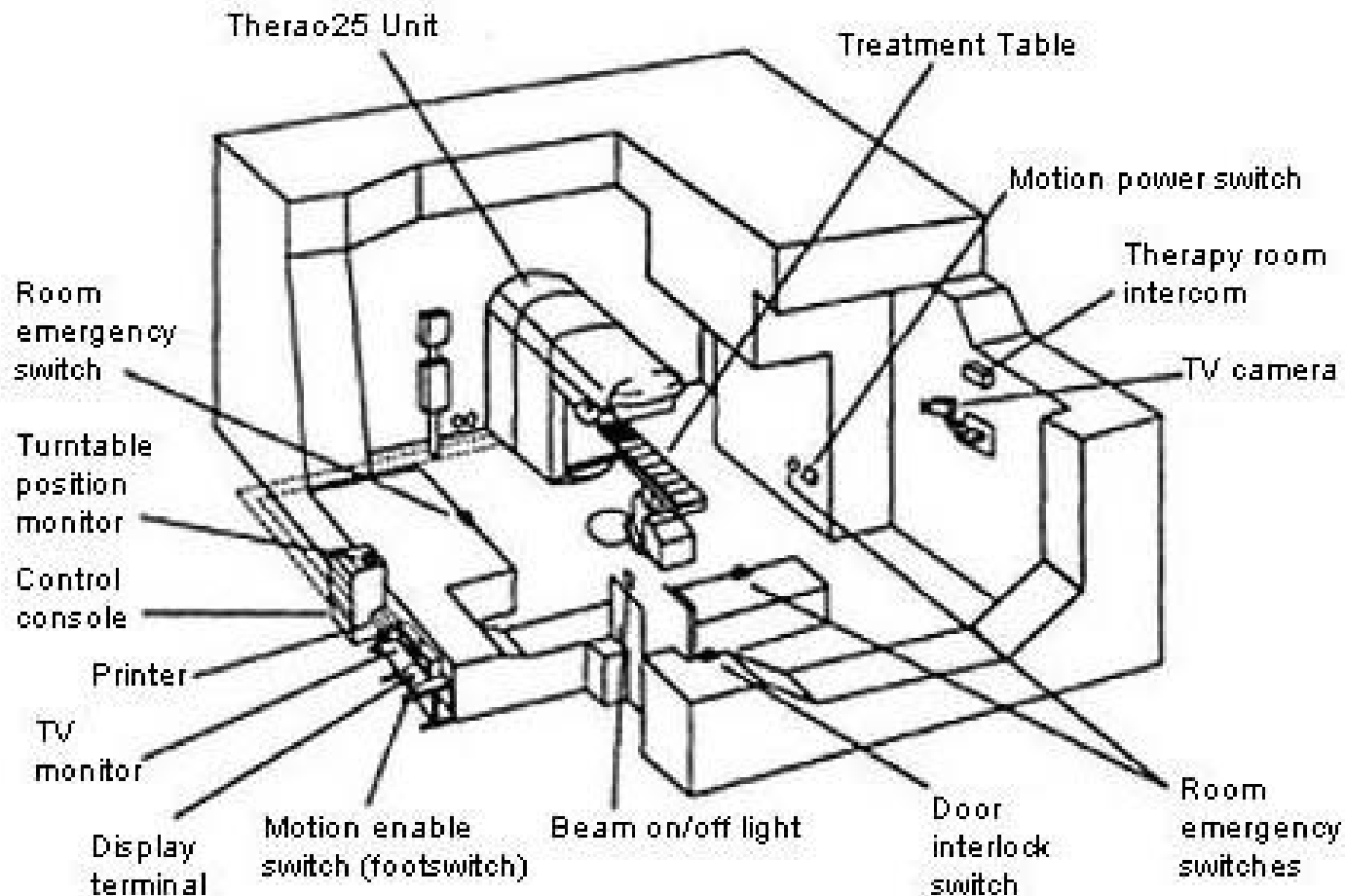


<http://www.fas.org/spp/starwars/gao/im92026.htm>

Problema e motivação



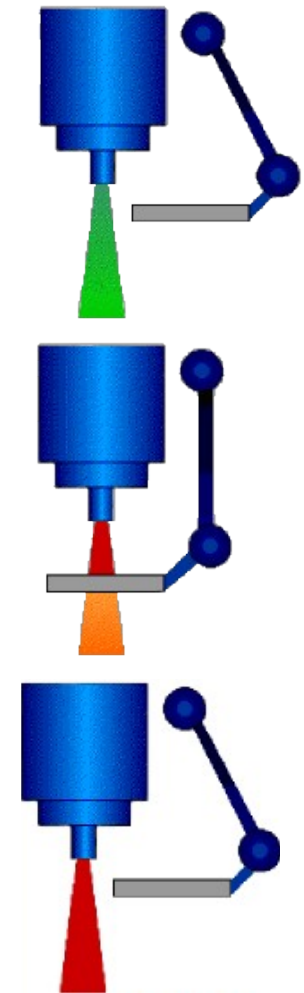
- Overdose de radiação em tratamento (1985–1987, 2001)
 - Causa: Mudança de hardware e erro no software.
 - Custo: 6 incidentes, 25 incidentes, 3 mortes confirmadas



http://computingcases.org/case_materials/therac/analysis/SocioTechnical_Analysis.html

24/06/2013

Rita de Cássia Cazu Soldi (<http://www.lisha.ufsc.br/rita>)



Problema e motivação



■ Ruptura em oleoduto (1999)

- **Causa:** Sistema inoperante não deixou o operador ver qual oleoduto estava rompido. Vazaram 237mil litros de gasolina.
- **Custo:** 3 mortos, 8 feridos, US\$45 milhões



Pipeline Accident Report: PAR-12-01 - <http://www.nts.gov/investigations/summary/PAR1201.html>

Algumas soluções...



■ Statical debugging

- **O que:** Reduzir o espaço de busca por um erro.
- **Como:** Utilizam estatísticas relacionadas ao fracasso para pode reduzir o conjunto de verificação.
- **Vantagens:**
 - É possível diminuir a quantidade de caminhos que levam ao erro.
 - Retorna pedaços de código em que existe uma probabilidade de serem origem de erros.
- **Problemas:**
 - Precisa de um grande conjunto de dados para realizar essa estatística adequadamente.
 - Retorna localizações espalhadas pelo código.

■ Program Slicing

- **O que:** Dividir para conquistar
- **Como:** Dividir o código em partes até conseguir isolar e remover os caminhos que **não** levam ao erro.
- **Vantagens:**
 - É necessário apenas um caminho que leva ao erro para a comparação.
 - É possível descobrir que algumas partes do código não geram determinado erro.
- **Problemas:**
 - Mesmo removendo uma boa parte do código que não gera o erro, não isola completamente o erro.
 - Desenvolvedor deve testar todos os caminhos que sobraram.

Algumas soluções...



■ Delta Debugging

- O que: Verificação de hipóteses
- Como: Construir uma hipótese baseada nas mudanças entre as versões. Refinar/rejeitar dependendo do resultado do teste.
- Vantagens:
 - Necessário apenas um caminho que leva ao erro para a comparação.
 - Hipóteses descartadas não fazem parte do caminho que gerou o erro.
- Problemas:
 - Necessário um caminho que não leva ao erro para comparar as versões e gerar boas hipóteses.
 - Pode demorar muito para encontrar uma boa hipótese.

Algumas soluções...



■ Capture/Replay

- **O que:** Capturar execução e reproduzir o erro
- **Como:** O programa é executado, todas as operações são gravadas e depois executadas passo a passo.
- **Vantagens:**
 - Possibilidade de encontrar a operação que origina o erro.
 - Comparação entre o executado e o esperado pode dar dicas sobre como corrigir o erro.
- **Problemas:**
 - É necessário executar todos os caminhos/comunicações possíveis de um objeto para outro até encontrar o erro.
 - Talvez não seja possível repetir o ambiente que gerou o erro.

■ Justitia

- **O que:** Monitoramento e depuração das camadas de software de sistemas embarcados
- **Como:** Gera testes e emula aplicações de acordo com o tipo de interface definida no modelo de teste.
- **Vantagens:**
 - Verificar cada camada do sistema separadamente.
 - Ensaaios individuais ou por grupos de interfaces.
- **Problemas:**
 - Depende da separação entre as camadas do sistema.
 - Teste de convergência precisa que todas as interfaces e funções devem ser executadas pelo menos uma vez.

■ ATEMES

- **O que:** Teste e depuração de software para sistemas embarcados de múltiplos núcleos.
- **Como:** Análise de código, geração de casos de teste e simulação da execução.
- **Vantagens:**
 - Quantidade de tipos de teste suportado (unitário, performance, etc.).
 - Geração de casos de teste.
 - Suporta instrumentação do código fonte.
- **Problemas:**
 - Teste unitário exige intervenção manual.
 - Verificação da cobertura de testes ainda não está completa.
 - Não fica clara a definição de “teste realizado com sucesso”.

- Melhorar o processo de depuração de software para sistemas embarcados.
 - Tempo;
 - Eficácia;
 - Qualidade;

- Reduzir a influência do hardware para a correta depuração do software para sistemas embarcados.
 - Configuração correta do software;

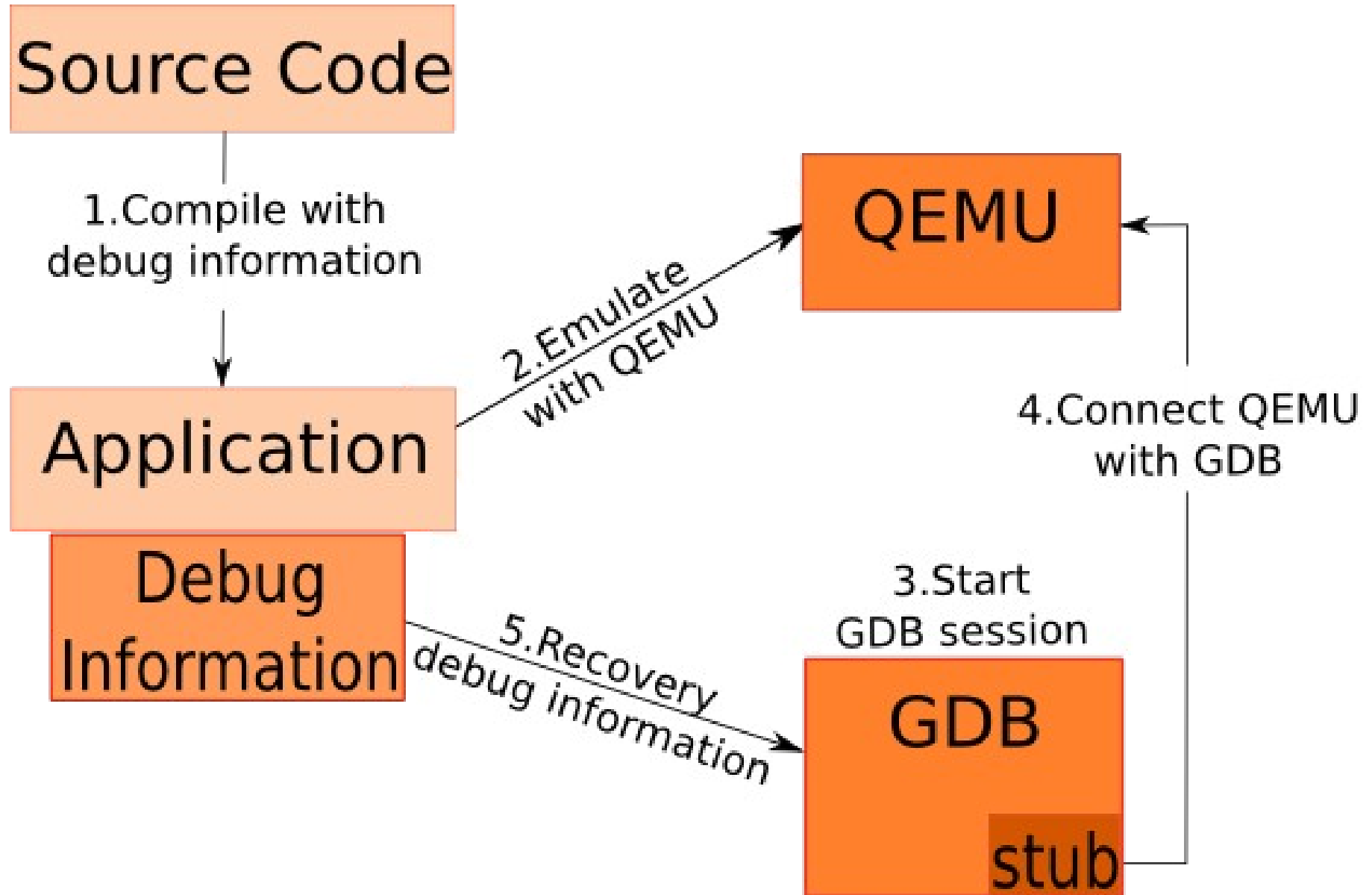
- A automação da depuração de software pode diminuir o *tempo* gasto e aumentar a *eficácia* da atividade de teste de sistemas.
- Uma correta configuração dos parâmetros e restrições do software é capaz de prevenir alguns tipos de erros e melhorar a *qualidade* do software.
- Simular o hardware e emular uma integração software/hardware pode ser uma alternativa para reduzir a *influência* deste componente no teste do software.

- Desenvolver uma ferramenta de automação de teste e depuração em sistemas embarcados.
 - Capaz de verificar se uma determinada configuração encontra-se em concordância com a especificação.
 - Deve ser possível analisar o estado atual do sistema sob teste para verificar se há algum erro durante alguma etapa do processo.
 - É desejável que a depuração utilize um hardware emulado ao invés do componente real.

- Experimentos:
 - Emular o Hardware sem interferência humana.
 - Troca automática de parâmetros do sistema.
 - Importar configuração para a troca de parâmetros.
 - Exportar configuração de parâmetros do sistema.
 - Sugestão de valores para a configuração dos parâmetros do sistema.
- Integração dos experimentos em uma única ferramenta.

Experimento

Emular o Hardware sem interferência humana



Experimento

Emular o Hardware sem interferência humana



■ QEMU

- Possibilidade de executar aplicações feitas para uma determinada máquina sob uma máquina diferente usando tradução dinâmica.
- Suporte a um conjunto nativo de máquinas-alvo e a potencial integração de uma nova máquina para este conjunto.

■ GDB

- Necessidade de observar o que ocorre dentro da aplicação enquanto ela é executada.
- Permite a verificação e o controle da execução do programa.
- Opção de realizar uma depuração remota.

Experimento

Emular o Hardware sem interferência humana



```
celo@ubuntu:~/Development/openepos/openepos/trunk$ qemu -fda img/periodic_thread_test.img -serial stdio -no-reboot -s -S  
open /dev/kvm: No such file or directory  
Could not initialize KVM, will disable KVM support  
pci_add_option_rom: failed to find romfile "pxe-rtl8139.bin"
```

A screenshot of a QEMU window title bar. It shows standard Linux window controls (close, maximize, and a disabled icon) followed by the text "QEMU [Stopped]". The text "[Stopped]" is highlighted with a yellow box.

QEMU [Stopped]

Experimento

Emular o Hardware sem interferência humana



```
Thread(entry=0x00008028,state=2,rank=0,stack={b=0x003fbfc8,s=16384},context={b=0x003fff9c,{eflags=0x00000200,eax=0,ebx=0,ecx=0,edx=0,esi=0,edi=0,ebp=0x00000000,esp=0x00000000,eip=0x00008028,cs=8,ds=16,es=16,fs=16,gs=16,ss=16,pdp=0x01ffc000}}) => 0x003fffc
Scheduler[chosen=0x00000000>::insert(0x003fffc)
Thread::reschedule()
Scheduler[chosen=0x003fffc>::choose() => 0x003fffc
Heap::alloc(this=0x004000dc,bytes=40) => 0x003fbf9c
Heap::alloc(this=0x004000dc,bytes=16388) => 0x003f7f98
Thread(entry=0x0000e500,state=1,rank=2147483647,stack={b=0x003f7f9c,s=16384},context={b=0x003fbf70,{eflags=0x00000200,eax=0,ebx=0,ecx=0,edx=0,esi=0,edi=0,ebp=0x00000000,esp=0x00000000,eip=0x0000e500,cs=8,ds=16,es=16,fs=16,gs=16,ss=16,pdp=0x01ffc000}}) => 0x003fbfa0
Scheduler[chosen=0x003fffc>::insert(0x003fbfa0)
Thread::reschedule()
Scheduler[chosen=0x003fffc>::choose() => 0x003fffc
Dispatching the first thread: 0x003fffc
```

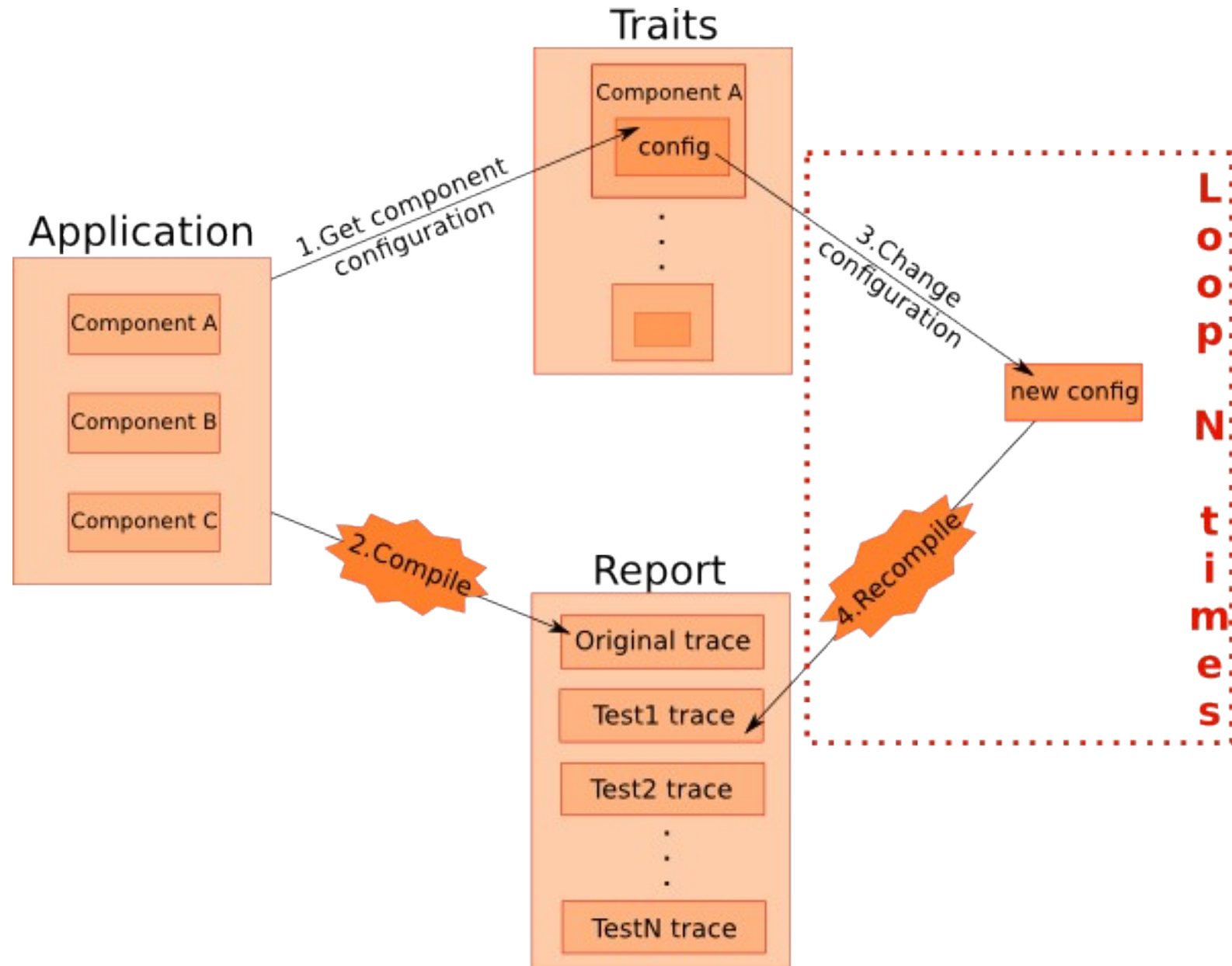
```
QEMU [Stopped]
Starting SeaBIOS (version pre-0.6.1-20100702_143500-palmer)
Booting from Hard Disk...
Boot failed: could not read the boot disk
Booting from Floppy...
Loading EPOS .... done;
This is EPOS;
```

```
celo@ubuntu:~/Development/openepos/openepos/trunk$ gdb
GNU gdb (GDB) 7.2-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
(gdb) target remote :1234
Remote debugging using :1234
0x0000fff0 in ?? ()
(gdb) file app/periodic_thread_test
A program is being debugged already.
Are you sure you want to change the file? (y or n) y
Reading symbols from /home/celo/Development/openepos/openepos/trunk/app/periodic_thread_test...done.
(gdb) b ma
machine function      main      max(long, long, long)
(gdb) b main
Breakpoint 1 at 0x8910
(gdb) b func_b()
Breakpoint 2 at 0x87d0
(gdb) continue
Continuing.

Breakpoint 1, 0x00008910 in main ()
(gdb)
```


Experimento

Troca automática de parâmetros de configuração



Experimento

Troca automática de parâmetros de configuração



- Requisito: Sistema com modelagem baseada em features e parametrização.
- Tipos de troca de configuração:
 - Totalmente aleatória.
 - Parcialmente aleatória.
 - Determinada pelo usuário.
- O que é definido como sucesso de troca de parâmetros?
 - Não há erros de compilação.
 - Há um registro da troca efetuada no relatório.
 - Diferença entre os traces

Experimento

Troca automática de parâmetros de configuração



```
Scheduler[chosen=0x003fffcc]::choose() => 0x003fffcc
Semaphore(value=0) => 0x003ffe58
Alarm(t=400000,tk=400,h=0x003ffe68,x=10) => 0x003ffe70
Thread::resume(this=0x003ffe34)
Scheduler[chosen=0x003fffcc]::resume(0x003ffe34)
Thread::reschedule()
Scheduler[chosen=0x003fffcc]::choose() => 0x003fffcc
Threads have been created. I'll wait for them to finish..
Thread::join(this=0x003fffee4,state=1)
Thread::suspend(this=0x003fffcc)
Scheduler[chosen=0x003fffcc]::suspend(0x003fffcc)
Thread::dispatch(prev=0x003fffcc,next=0x003fffee4)
-----A Semaphore::p(th
Thread::sleep(running=0x003fffee4,q=0x003fff08)
Scheduler[chosen=0x003fffee4]::suspend(0x003fffee4)
Thread::dispatch(prev=0x003fffee4,next=0x003ffe8c)
-----B Semaphore::p(th
Thread::sleep(running=0x003ffe8c,q=0x003ffeb0)
Scheduler[chosen=0x003ffe8c]::suspend(0x003ffe8c)
Thread::dispatch(prev=0x003ffe8c,next=0x003ffe34)

QEMU: Terminated via GDBstub
celo@ubuntu:~/Development/openepos/openepos/trunk$
```

```
celo@ubuntu: ~/Development/openepos/openepos/trunk
File Edit View Search Terminal Help
celo@ubuntu:~$ cd $EPOS && gdb -batch -x gdb_script.sh app/periodic_thread_test
0x0000fff0 in System::Thread::~Thread() ()
Breakpoint 1 at 0x8910
Breakpoint 2 at 0x8550
Breakpoint 3 at 0x87d0
Breakpoint 4 at 0x8690

Breakpoint 1, 0x00008910 in main ()
Breakpoint 2, 0x00008550 in func_a() ()
Breakpoint 3, 0x000087d0 in func_b() ()
Breakpoint 4, 0x00008690 in func_c() ()
celo@ubuntu:~/Development/openepos/openepos/trunk$
```

Experimento

Troca automática de parâmetros de configuração



```
.*.*.*.*.* Test Report *.*.*.*.*.*  
Application= dmec_app  
  
Original line = #define NUM_WORKERS 6  
VALUES = 67,53,87,3,64,35,16,75,82,47,  
79,70,81,12,46,84,68,18,76,26,  
86,66,90,89,67,9,87,19,81,24,  
31,2,12,24,58,33,15,3,55,4,  
0,17,67,96,0,34,5,70,34,35,  
27,41,40,88,94,45,96,7,55,72,  
98,42,91,97,4,70,28,35,69,29,  
34,19,28,72,15,96,29,39,87,72,  
27,15,23,10,92,72,8,12,17,40,  
62,42,17,90,45,83,35,81,10,7
```

Experimento

Importar uma configuração inicial para a troca de parâmetros



```
<test>
  <application name="philosopher_dinner_app"></application>
    <configuration>
      <trait>
        <id>ARCH</id>
        <value>IA32</value>
        <value>AVR8</value>
      </trait>

      <debug>
        <path>"/home/breakpoint_philosopher.txt"</path>
      </debug>
    </configuration>
</test>
```

Experimento

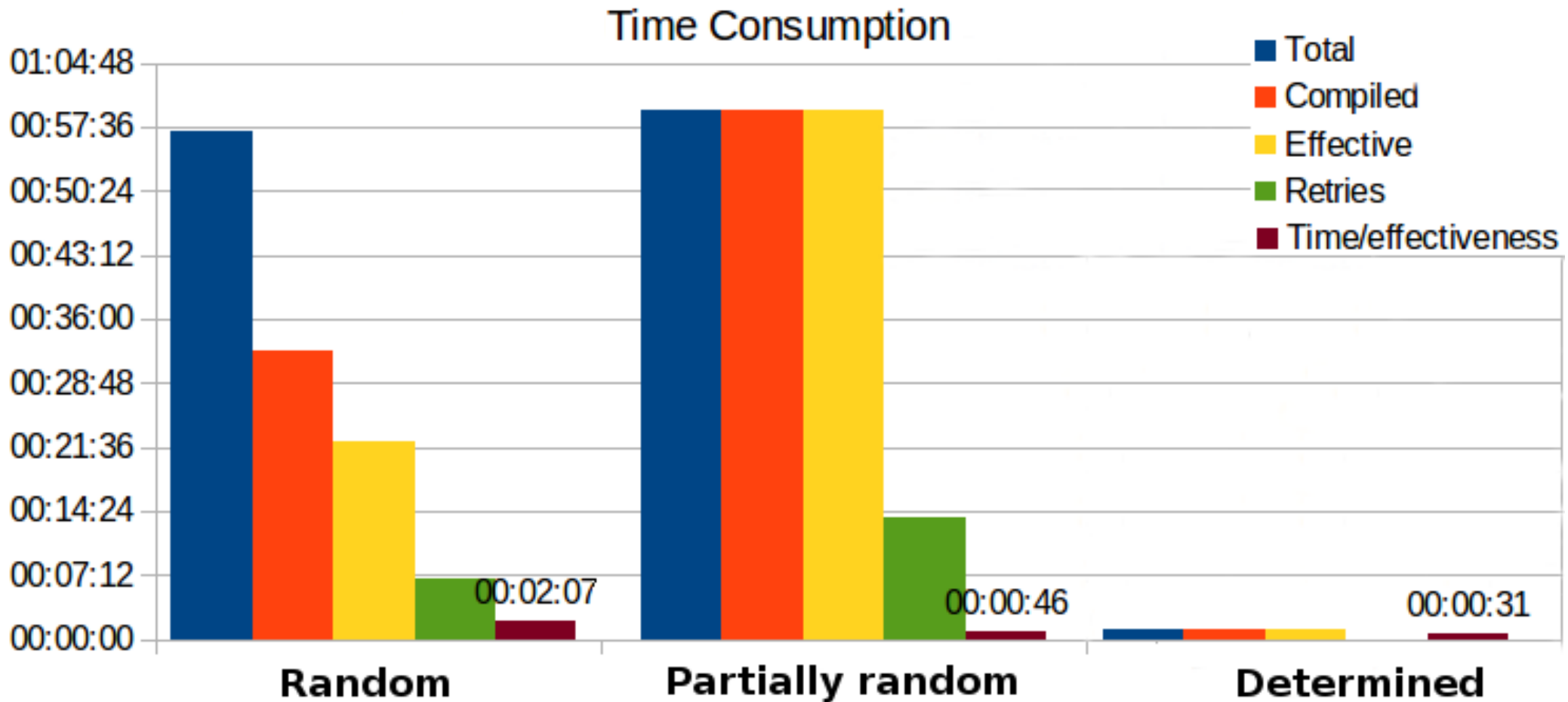
Importar uma configuração inicial para a troca de parâmetros



- Configurações importadas através do XML:
 - Aplicação
 - Nome;
 - Troca de parâmetros
 - Traits;
 - Valores;
 - Intervalos;
 - Número de tentativas;
 - Depuração
 - Arquivo de depuração;
 - Comparação entre traces;

Experimento

Integração dos experimentos e alguns resultados



■ Prática:

- Experimentos:
 - Exportar configuração de parâmetros do sistema.
 - Sugestão de valores para a configuração dos parâmetros do sistema.
 - Integração dos experimentos.
- Extrair métricas.

■ Teórica:

- Produção do texto da dissertação de mestrado.
- Produção e submissão de artigos.

Cronograma



1. Exportar configuração de parâmetros do sistema.

Atividade	Jun 2013	Jul 2013	Ago 2013	Set 2013	Out 2013	Nov 2013	Dez 2013
1	X						
2.1		X					
2.2		X					
2.3			X				
3.1			X				
3.2			X	X			
3.3				X			
4			X	X	X	X	X
5	X	X	X	X	X	X	X
6							X

Cronograma



2. Sugestão de valores para a configuração dos parâmetros do sistema.

2.1. Pesquisar algoritmos/técnicas de sugestão de configurações ideais.

Atividade	Jun 2013	Jul 2013	Ago 2013	Set 2013	Out 2013	Nov 2013	Dez 2013
1	X						
2.1		X					
2.2		X					
2.3			X				
3.1			X				
3.2			X	X			
3.3				X			
4			X	X	X	X	X
5	X	X	X	X	X	X	X
6							X

Cronograma



2. Sugestão de valores para a configuração dos parâmetros do sistema.

2.2. Implementar/implantar o algoritmo selecionado.

Atividade	Jun 2013	Jul 2013	Ago 2013	Set 2013	Out 2013	Nov 2013	Dez 2013
1	X						
2.1		X					
2.2		X					
2.3			X				
3.1			X				
3.2			X	X			
3.3				X			
4			X	X	X	X	X
5	X	X	X	X	X	X	X
6							X

Cronograma



2. Sugestão de valores para a configuração dos parâmetros do sistema.

2.3. Realimentação do algoritmo com a configuração sugerida.

Atividade	Jun 2013	Jul 2013	Ago 2013	Set 2013	Out 2013	Nov 2013	Dez 2013
1	X						
2.1		X					
2.2		X					
2.3			X				
3.1			X				
3.2			X	X			
3.3				X			
4			X	X	X	X	X
5	X	X	X	X	X	X	X
6							X

Cronograma



3. Avaliação do trabalho.

3.1. Pesquisar métricas de qualidade de software de sistemas embarcados.

Atividade	Jun 2013	Jul 2013	Ago 2013	Set 2013	Out 2013	Nov 2013	Dez 2013
1	X						
2.1		X					
2.2		X					
2.3			X				
3.1			X				
3.2			X	X			
3.3				X			
4			X	X	X	X	X
5	X	X	X	X	X	X	X
6							X

Cronograma



3. Avaliação do trabalho.

3.2. Especificar quais métricas serão utilizadas no meu trabalho.

Atividade	Jun 2013	Jul 2013	Ago 2013	Set 2013	Out 2013	Nov 2013	Dez 2013
1	X						
2.1		X					
2.2		X					
2.3			X				
3.1			X				
3.2			X	X			
3.3				X			
4			X	X	X	X	X
5	X	X	X	X	X	X	X
6							X

3. Avaliação do trabalho

3.3. Avaliar o trabalho utilizando as métricas definidas anteriormente.

Atividade	Jun 2013	Jul 2013	Ago 2013	Set 2013	Out 2013	Nov 2013	Dez 2013
1	X						
2.1		X					
2.2		X					
2.3			X				
3.1			X				
3.2			X	X			
3.3				X			
4			X	X	X	X	X
5	X	X	X	X	X	X	X
6							X

Cronograma



4. Produção do texto da dissertação.

Atividade	Jun 2013	Jul 2013	Ago 2013	Set 2013	Out 2013	Nov 2013	Dez 2013
1	X						
2.1		X					
2.2		X					
2.3			X				
3.1			X				
3.2			X	X			
3.3				X			
4			X	X	X	X	X
5	X	X	X	X	X	X	X
6							X

Cronograma



5. Produção e submissão de artigos

Atividade	Jun 2013	Jul 2013	Ago 2013	Set 2013	Out 2013	Nov 2013	Dez 2013
1	X						
2.1		X					
2.2		X					
2.3			X				
3.1			X				
3.2			X	X			
3.3				X			
4			X	X	X	X	X
5	X	X	X	X	X	X	X
6							X

Cronograma



6. Defesa da dissertação de mestrado.

Atividade	Jun 2013	Jul 2013	Ago 2013	Set 2013	Out 2013	Nov 2013	Dez 2013
1	X						
2.1		X					
2.2		X					
2.3			X				
3.1			X				
3.2			X	X			
3.3				X			
4			X	X	X	X	X
5	X	X	X	X	X	X	X
6							X

■ Limitações:

- Não possuo dedicação exclusiva, trabalho no mestrado aproximadamente 15 horas semanais.
- Acordo de liberação da empresa durante uma tarde por mês para reunião com orientador.
- Prazo para finalização do mestrado é dezembro de 2013.

■ Artigos:

- Submetidos:
 - ASE (A1) – Automated Software Engineering - 24/07
 - EMSOFT (A2) – International Conference on Embedded Software - 05/07
 - EUC (B2) – International Conference on Embedded and Ubiquitous Computing – 15/08
- Em andamento:
 - DATE (A1) - Design, Automation and Test in Europe
 - SBESC (B4) - Simpósio Brasileiro de Engenharia de Sistemas Computacionais
 - ETS (B2) - European Test Symposium
 - LATW (B4) - Latin American Test Workshop

Obrigada.

Perguntas?