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

- O custo de erros de software[1]:
  - Falha no sistema de defesa contra mísseis: 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.
  - Overdose de radiação em tratamento. (1985 – 1987, 2001)
    - Causa: Mudança de hardware. Hardware anterior limitava o número da dosagem, o novo estimou dose 100 vezes maior.
    - Custo: 6 incidentes, 25 incidentes, 3 mortes confirmadas.
  - 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

[1] Zhivich, M.; Cunningham, R.K., "The Real Cost of Software Errors," Security & Privacy, IEEE , vol.7, no.2, pp.87,90, March-April 2009.

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.

Algumas soluções...

- 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.

Algumas soluções...

- 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.
    - Ensaios 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.

Algumas soluções...

- 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”.

Objetivo

- 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;

Hipótese

- A automação da depuração de software pode dimimuir 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 previnir alguns tipos de erros e melhorar a *qualidade* do software.

- Simular o hardware e emular uma integração software/hardware pode ser uma alterativa para reduzir a *influência* deste componente no teste do software.

Proposta

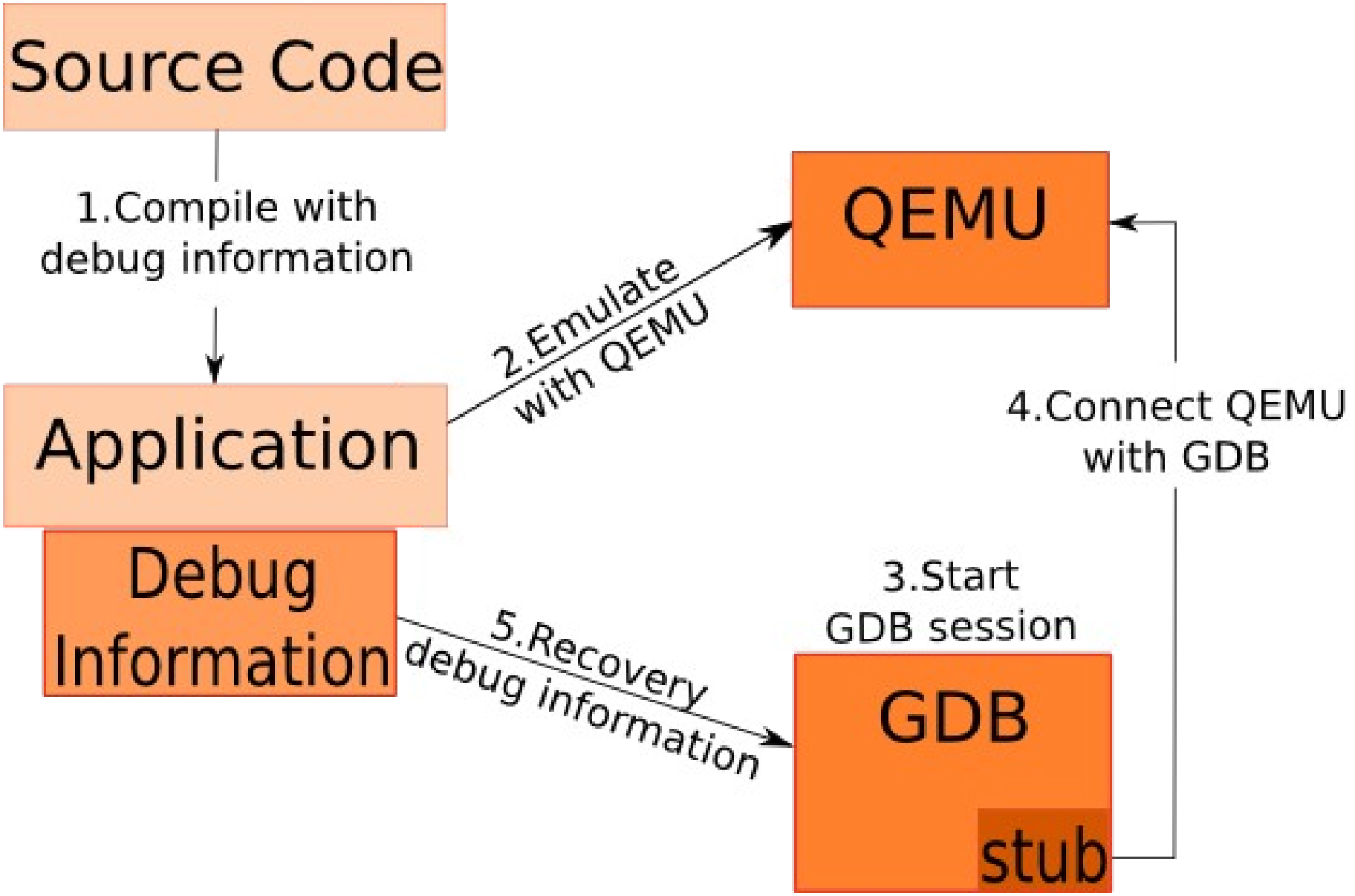
- Desenvolver uma ferramenta de automação de teste e depuração em sistemas embarcados.
  - Capaz de verificar se uma determinada configuração encontra-se em concordância com a espeficicação.
  - Deve ser possível analisar o estado atual do sistema sob teste para verificar se há algum erro durante alguma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

Metodologia

- Experimentos:
  - Emular o Hardware sem interferência humana.
  - Troca automática de parâmetros do sistema.
  - Importar configuração para a troca de parâmentros.
  - 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



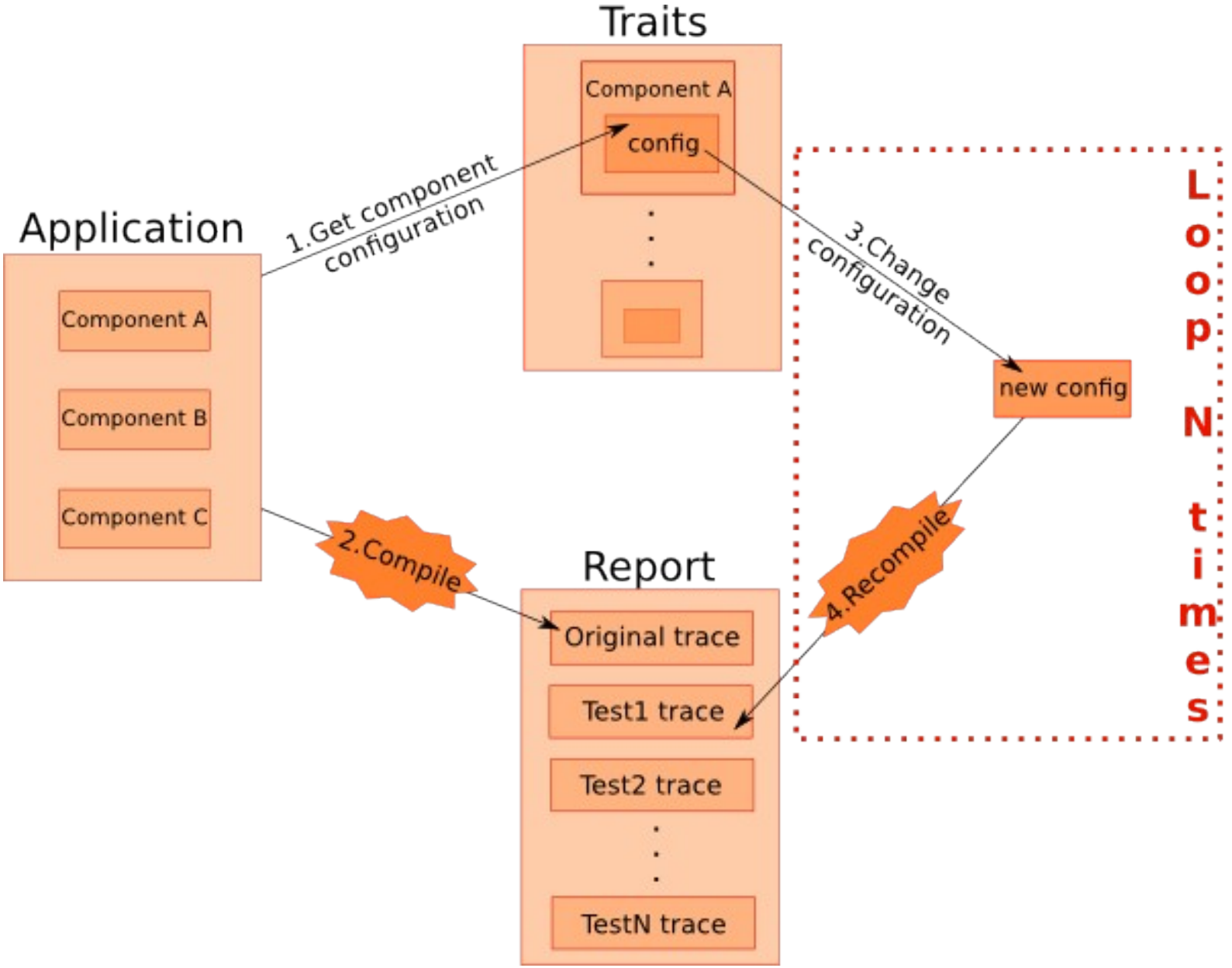
Experimento
Emular o Hardware sem interferência humana

- 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

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

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;

Cronograma

- 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

Cronograma

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**

Cronograma

- 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), EMSOFT (A2), EUC (B2).
  - Em andamento: DATE (A1), SBESC (B4).
  - Sem data: ETS (B2), LATW (B4).

Obrigada.

Perguntas?