Using Background Colors to Support Program Comprehension in Software Product Lines

Janet Feigenspan, Michael Schulze
Maria Papendieck, Christian Kästner, Raimund Dachselt, Veit Köppen, Mathias Frisch

EASE 2011

Introdução

- SPLs: mecanismo eficiente para implementar software variável

- Geralmente implementadas com preprocessadores (#ifdefs)
  - Simples de usar
  - Flexível e expressivo
  - Já está integrado em várias linguagens

Introdução

- Código complexo e ofuscado.
- Dificuldade de compreensão
- Custo de manutenção elevado
- *#ifdef hell*

Introdução

- Solução: AOP, componentes

Mas...

- Introdução de novos conceitos no mercado é difícil
  - Gasta muito tempo
  - Muito código legado

Introdução

- Como melhorar os preprocessadores?
- Durante manutenção, um programador gasta em média 50-60% do tempo entendendo o código.
- Manutenção é o maior custo no desenvolvimento de software
- Usar cores para melhorar compreensão do código

O Problema

- Trechos muito longos de código podem ser anotados com #ifdefs.
  - Geralmente o #ifdef não aprece na mesma tela que o #endif correspondente.

```
 static int __rep_queue_filedone(dbenv, rep, rfp)
   DB_ENV *dbenv;
   REP *rep;
   __rep_fileinfo_args *rfp; {
#ifndef HAVE_QUEUE
   COMPQUIET(rep, NULL);
   COMPQUIET(rfp, NULL);
   return (__db_no_queue_am(dbenv));
#else
 db_pgno_t first, last;
 u_int32_t flags;
   int empty, ret, t_ret;
#ifdef DIAGNOSTIC
   DB_MSGBUF mb;
#endif
 // over 100 lines of additional code
#endif }
```

O Problema

- #ifdefs podem ser aninhados.
  - SPLs típicas podem ter até 9 #ifdefs aninhados.

- #ifdefs são texto.
  - Não diferem do resto do código.

O Problema

- O uso de cores pode melhorar o entendimento do programa.
  - Experimento anterior com SPL de tamanho médio
    - 5000 LOC
    - 4 features
  - 1 cor para cada feature
  - Cores melhoraram o entendimento do programa para alguns tipos de tarefas.

O Problema

- O uso de cores é escalável para grandes SPLs do mercado?
  - Com 4 features é fácil fazer o mapeamento 1x1 de cores e features.
  - Não é à toa que o resultado foi positivo.

O Problema

- Escalabilidade é questionada por dois motivos:
  - A capacidade de trabalho da memória humana é limitada.
    - Aproximadamente 7±2 itens são armazenados na memória humana.
  - Habilidade humana em distinguir cores é limitada
    - Ser humano consegue distinguir 2 milhões de cores, com comparação direta.
    - Sem comparação direta, somente algumas cores.

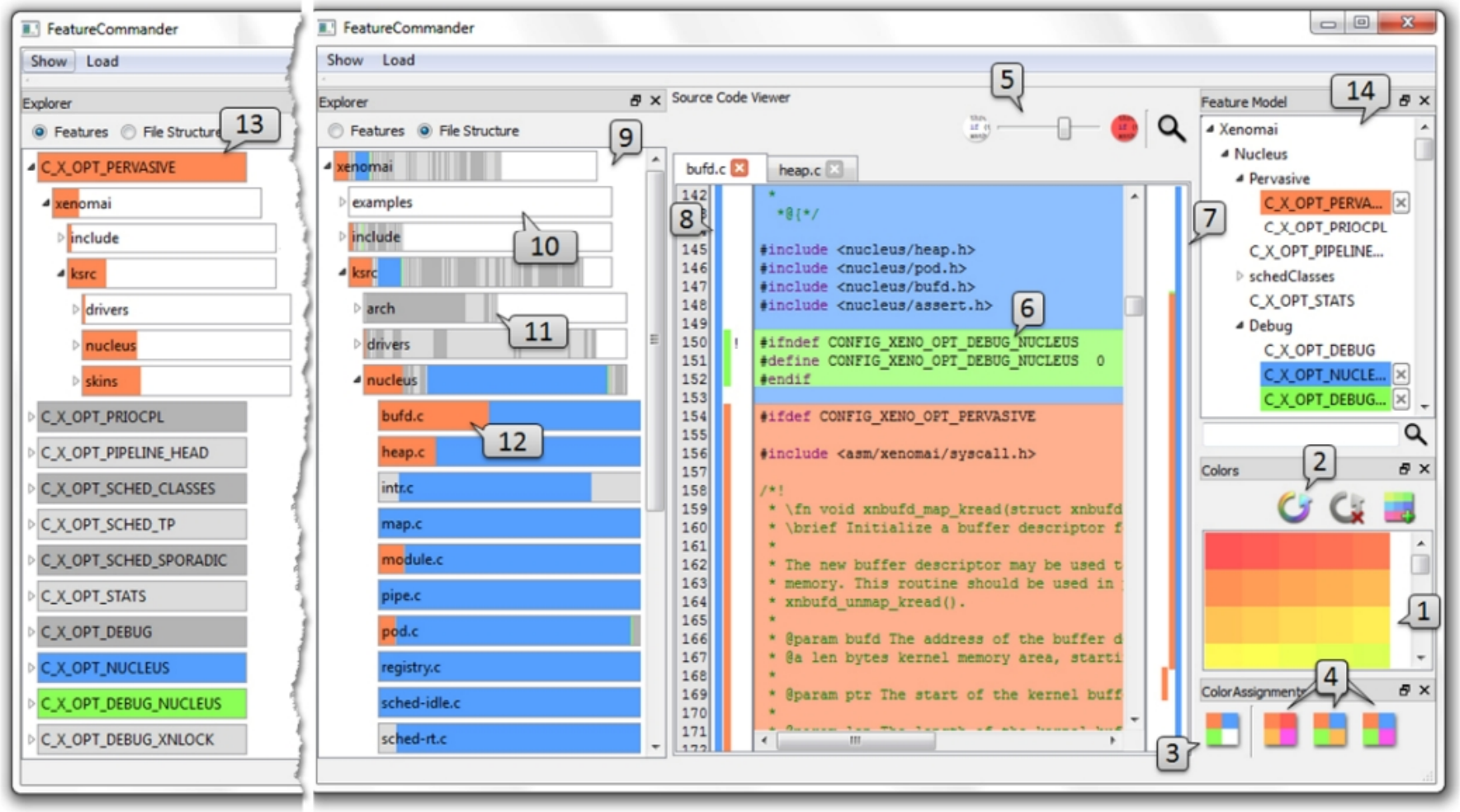
O Problema

- Mapeamento 1x1 entre cores e features é inviável em grandes SPLs com centenas de features.
- Na maior parte do código, somente 3 features estão presentes no código sendo mostrado na tela.
- Bugs podem ser restritos a uma ou poucas features.

FeatureCommander

- Ferramenta para tornar viável o desenvolvimento de grandes SPLs utilizando cores.
  - Possui múltiplas visualizações das features.
  - Cores podem ser atribuídas às features.
  - Atribuição de cores pode ser salva e recarregada posteriormente.
  - Features sem cor são cinza em todas as visualizações.

FeatureCommander



FeatureCommander

- Memória humana limitada
  - Atribuição customizável de cores
- Capacidade limitada de distinguir cores
  - Poucas cores presentes simultaneamente no código.

FeatureCommander

- Parece que resolve o problema.
- É realmente escalável?

Experimento

Objetivo

- Hipótese 1:

Cores melhoram o entendimento de grandes SPLs

- Hipótese 2:

Programadores preferem trabalhar com cores em grandes SPLs

Experimento

Material Experimental

- Xenomai: a real-time extension for Linux
  - 165082 LOC
  - 2 (???) Linhas de código de feature
  - 343 Features

- 2 versões modificadas do FeatureCommander
  - Com cores
  - Sem cores

Experimento

Material Experimental

- Questionário impresso
  - Dificuldade de cada tarefa
  - Motivação para resolver a tarefa
  - Performance esperada

Experimento

Subjects

- 9 mestrandos e 5 doutorandos
  - Familiarizados com C
    - 4 em 5 na escala Likert
  - Homens
  - Nenhum daltônico

- Separados em dois grupos homogêneos

Experimento

Tarefas

- 10 tarefas, divididas em 3 tipos:
  - Aquecimento (2)
    - Para os programadores se familiarizarem com o experimento
  - Manutenção (2)
    - Localizar e resolver um *bug*.
  - Estática (6)
    - Examinar a estrutura do código

Experimento

Tarefas

- 3 tipos de tarefas estáticas:
  - Identificar todos os arquivos nos quais alguma feature foi implementada
  - Localizar #ifdefs aninhados
  - Identificar todas as features que aparecem em um determinado arquivo
- Ex:

  S1: In which files does feature CONFIG_XENO_OPT_STATS occur?

Experimento

Tarefas

- Bugs típicos de C e sistemas operacionais foram introduzidos para as tarefas de manutenção.
- Ex:

If the PEAK parallel port dongle driver (XENO_ DRIVERS_CAN_SJA1000_PEAK_DNG) should be unloaded, a segmentation fault is thrown. The problem occurs, when features CONFIG_XENO_DRIVERS_CAN and CONFIG_XENO_DRIVERS_CAN_SJA1000 and CONFIG_XENO_DRIVERS_CAN_SJA1000_PEAK_DNG are selected

Experimento

Design

- Pessoas foram divididas em dois grupos: A e B.
- Experimento foi feito em duas fases:
  - Fase 1: grupo A com cores e grupo B sem cores.
  - Fase 2: o inverso.
- Em cada fase as mesmas tarefas eram aplicadas para os dois grupos:
  - Fase 1: W1, S1, S2, S3, M1
  - Fase 2: W2, S4, S5, S6, M2

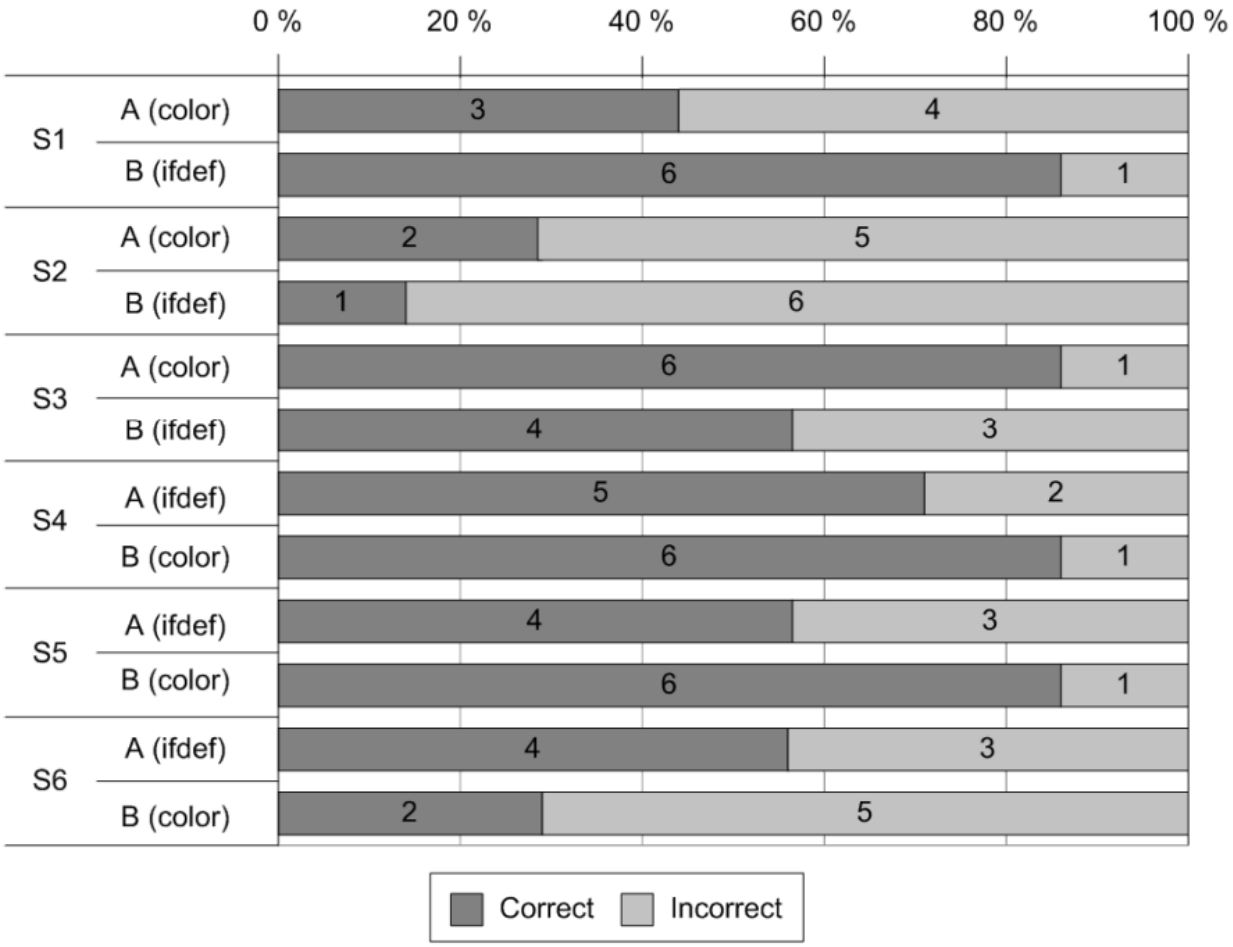
Experimento

Condução

- Monitores TFT-LCD 17"
- Ao terminar uma tarefa, a próxima era imediatamente iniciada.
- Ao final de cada fase um questionário foi passado
  - Dificuldade, motivação, performance esperada
- Ao final da segunda fase, foi perguntado qual versão (com ou sem cores) foi a preferida, e qual é mais adequada para trabalhar com #ifdefs

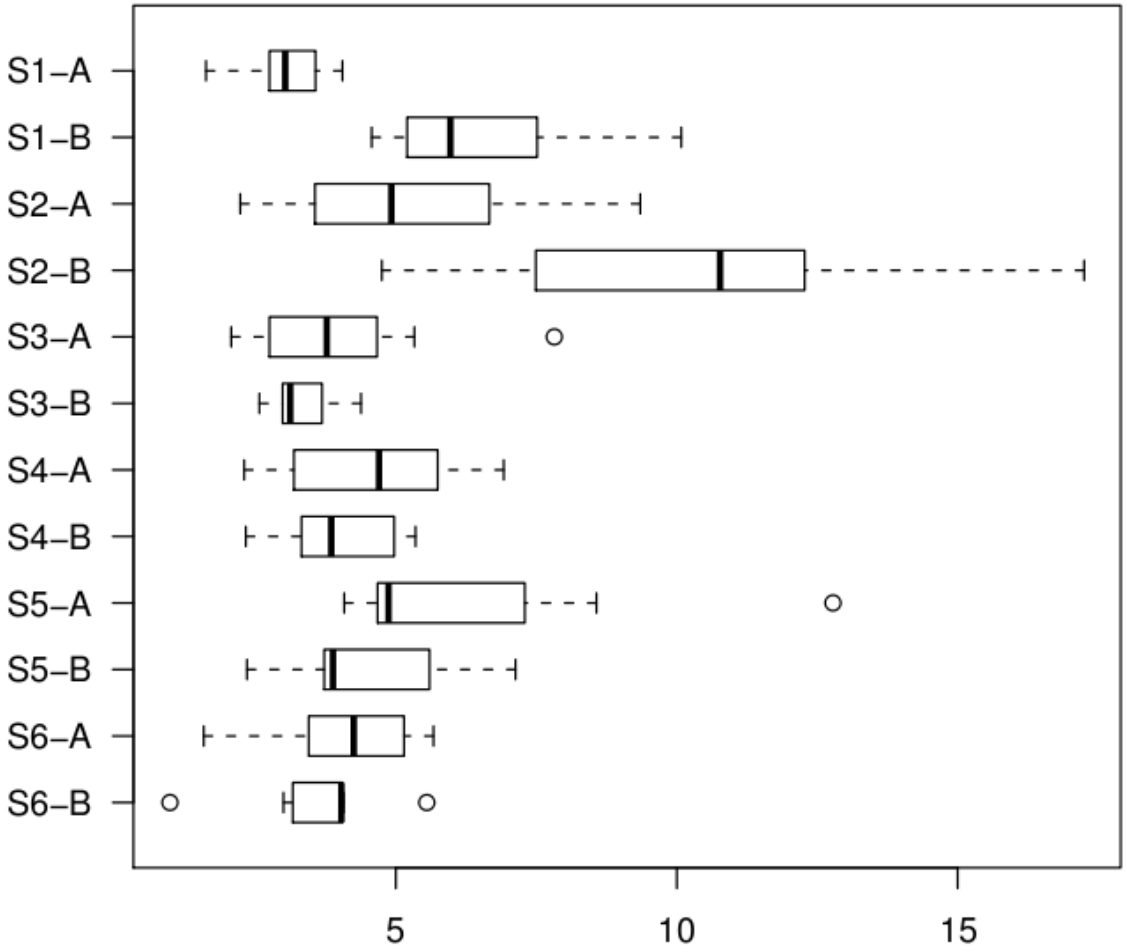
Resultados

- Respostas corretas



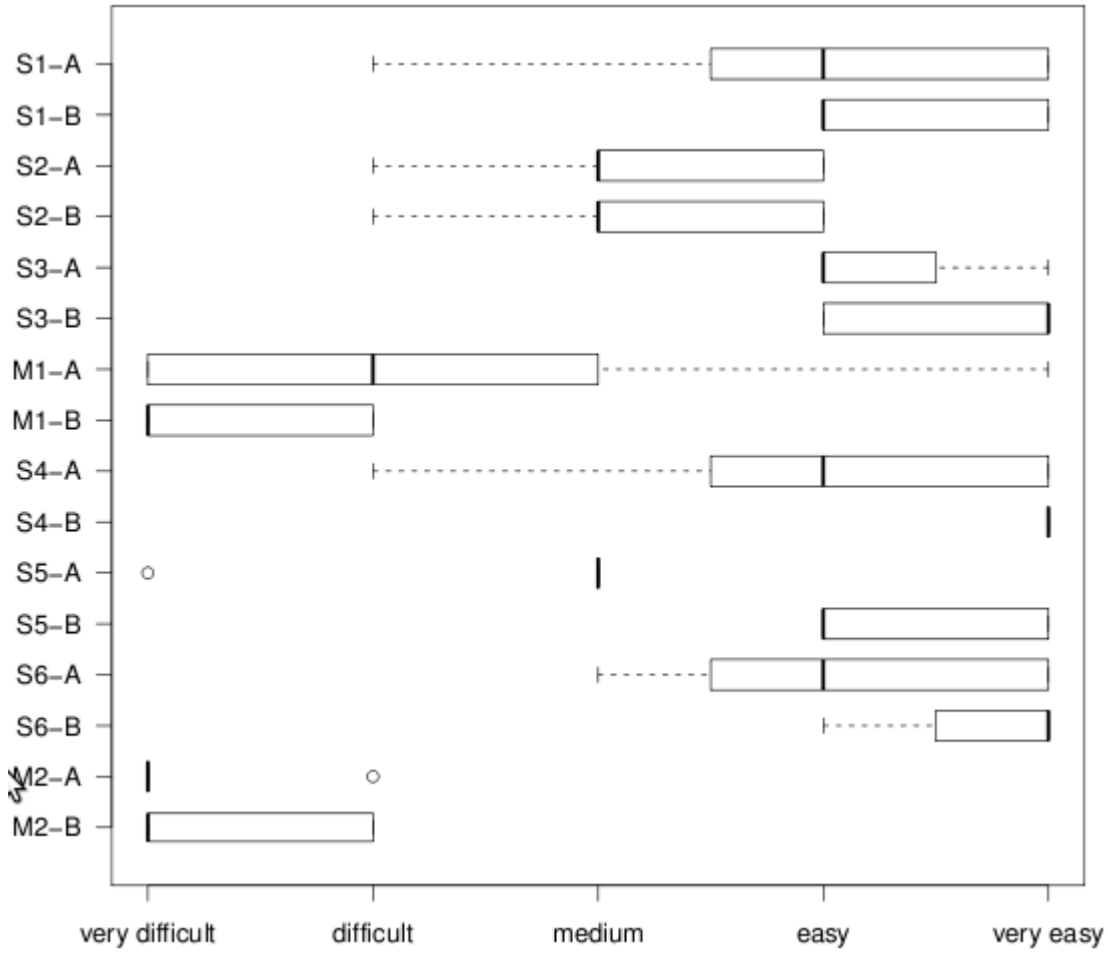
Resultados

- Tempo de resposta para tarefas estáticas



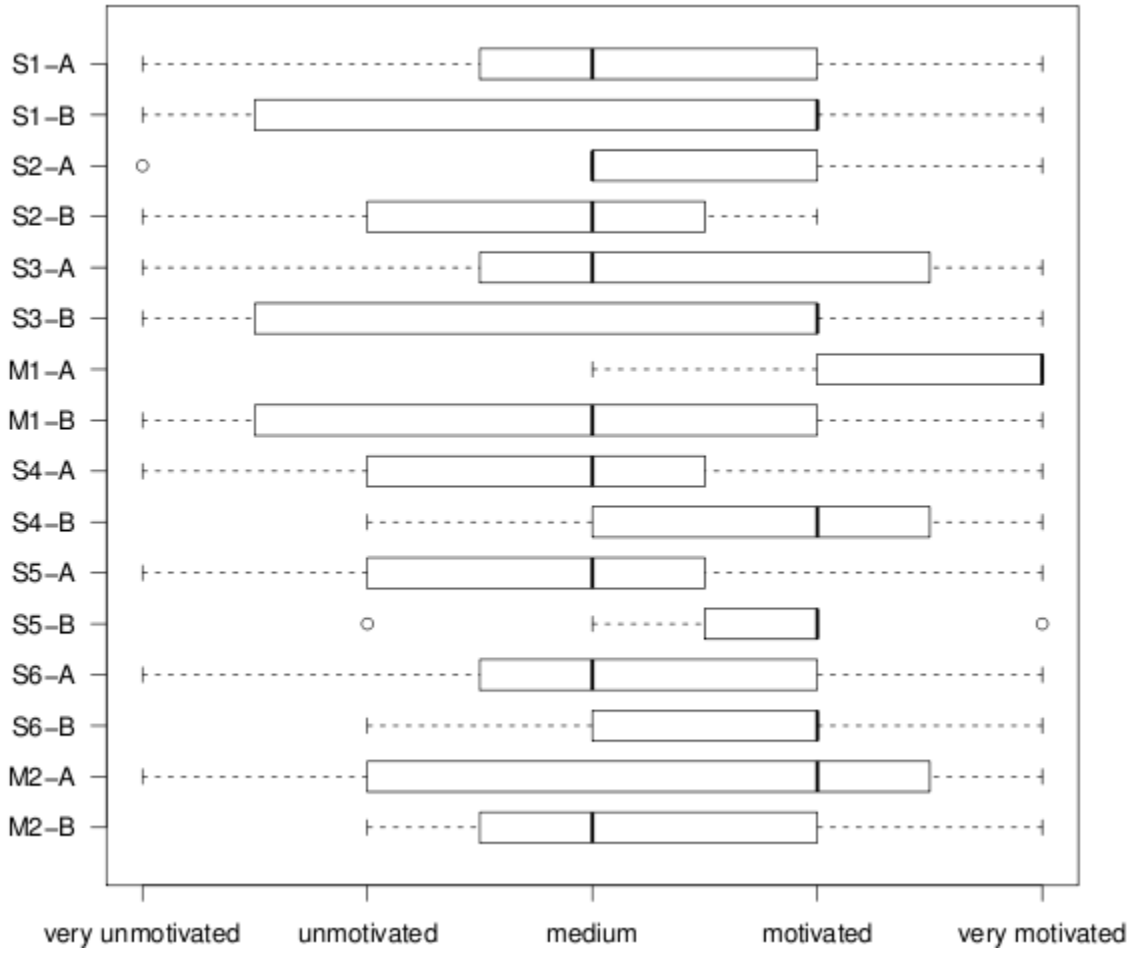
Resultados

- Opinião dos entrevistados sobre dificuldade



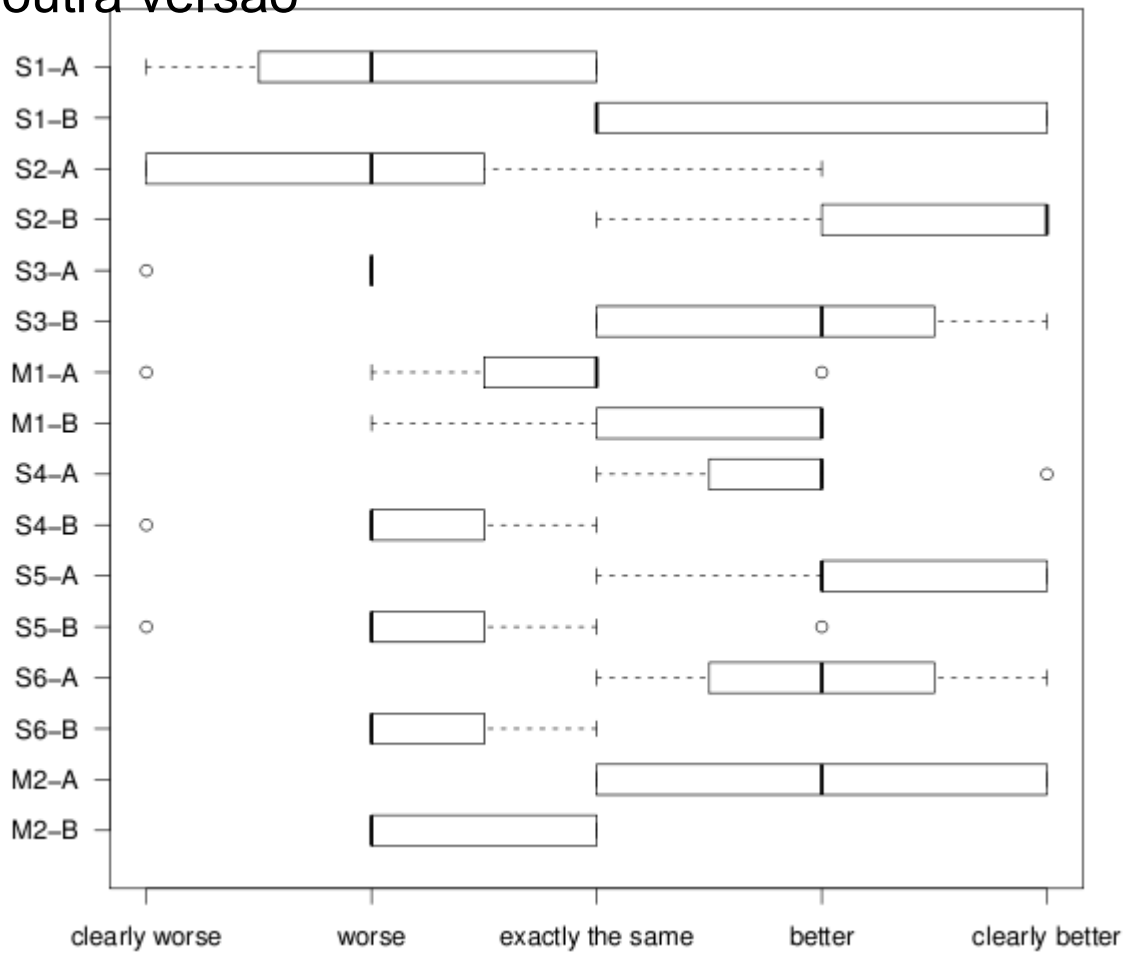
Resultados

- Opinião dos entrevistados sobre motivação



Resultados

- Opinião dos entrevistados sobre expectativa de performance com a outra versão



Análise

Teste de hipótese

- Para a respostas corretas foi utilizado o teste χ2:
  - Número de respostas corretas e erradas foram somados.
  - Foi feita uma comparação do número de respostas corretas e erradas entre as tarefas S1+S2+S3 e S4+S5+S6.

Análise

Teste de hipótese

- O teste χ2 não indicou nenhuma diferença significante no número de respostas corretas para as tarefas estáticas.

- Como ninguém acertou nenhuma tarefa de manutenção, não foi necessário fazer o teste para elas.

Análise

Teste de hipótese

- Para o tempo de resposta foram feitas várias comparações:
  - A x B
  - A (fase 1) x A (fase 2)
  - B (fase 1) x B (fase 2)

- Foi utilizada a correção de Bonferoni para achar um nível de significância entre as comparações.

Análise

Teste de hipótese

- A x B
  - Só houveram diferenças significativas nas tarefas S1 e S2 (versão com cores foi mais rápida).
  - Quando os grupos trocaram de versão, não houve diferenças.

Análise

Teste de hipótese

- A(1) x A(2) e B(1) x B(2)
  - Somente no grupo B houveram diferenças significativas
    - Tarefas S4 e S5 (com cores) foram mais rápidas que as tarefas S1 e S2 (sem cores).
  - Eram esperadas diferenças no grupo A, mas elas não ocorreram.
- Não é possível confirmar nem rejeitar a hipótese.

Análise

Teste de hipótese

- Para a opinião dos testadores foi utilizado o teste Mann-Whitney-U.
  - Dificuldade
    - Grupo B (com cores) achou as tarefas S4 e S5 muito mais fáceis que o grupo A (sem cores).
    - Consequentemente grupo B foi mais rápido na tarefa S4 comparada com a S1, e na S5 comparada com a S2.

Análise

Teste de hipótese

- Motivação
  - Diferença significante na motivação para as primeiras tarefas de manutenção.
  - Grupo A (com cores) estava muito mais motivado que o grupo B (sem cores).

Análise

Teste de hipótese

- Expectativa de performance com a outra versão
  - Diferença significativa em todas as tarefas (exceto M1).
  - Programadores acharam que foram piores quando trabalharam sem as cores.

Interpretação

H1: Cores melhoram o entendimento de grandes SPLs

- Não houveram diferenças significativas nas respostas corretas das tarefas estáticas.
- Na primeira fase, em duas tarefas estáticas, quem trabalhou com cores foi mais rápido.
- Na segunda fase não houveram diferenças significativas entre os dois grupos.
- Grupo B foi mais rápido nas tarefas S4 e S5 (com cores) do que nas tarefas S1 e S2 (sem cores).
- Em duas tarefas estáticas as cores melhoraram o entendimento do programa.

Interpretação

H1: Cores melhoram o entendimento de grandes SPLs

- Para a terceira tarefa (localizar todas as features em um arquivo) as cores não fizeram diferença.
  - 12 features
  - Excede a capacidade de trabalho da memória humana.
  - Em outras tarefas o máximo de cores simultâneas foi 9, o que está dentro do limite.
- Não é possível saber se esse resultado foi causado pelo excesso de cores, ou pela tarefa.

Interpretação

H1: Cores melhoram o entendimento de grandes SPLs

- Não houve diferença no tempo de resposta quando os programadores começaram com cores, e depois foram para a versão sem cores.
- Quando o oposto ocorre, o tempo de resposta melhora.

Interpretação

H1: Cores melhoram o entendimento de grandes SPLs

- Cores *podem* ajudar um programador a se familiarizar com grandes SPLs.
- Mesmo resultado obtido no experimento anterior.

Interpretação

H2: Pessoas preferem trabalhar com cores em grandes SPLs

- Quem trabalhou com cores na segunda fase achou as tarefas estáticas mais fáceis, comparado com o outro grupo.
  - Tarefas parecem que ficam mais fáceis com o uso das cores.
- Programadores acharam que foram melhores na versão com cores, do que na sem cores.

Interpretação

H2: Pessoas preferem trabalhar com cores em grandes SPLs

- Quase todos os programadores disseram que as cores eram mais adequadas para se trabalhar com grandes SPLs.

- Confirma a segunda hipótese.

Validação

- Não há uma forma padrão de avaliar a experiência em programação.
  - Não é possível saber o quão boa foi essa avaliação, na hora de formar os grupos.
  - Questionário foi baseado na literatura, e *experts* avaliaram o questionário.

Validação

- Ninguém resolveu as tarefas de manutenção.
  - Tarefas eram reais (feitas por um *expert* em C e Xenomai)
  - Não era o foco do experimento.
  - Outros estudos apontam que cores não ajudam neste tipo de tarefa.

Validação

- Amostra foi muito pequena, e a maior parte era de mestrandos (pouca experiência em programação)
  - Todos os programadores usaram as duas versões do FeatureCommander.
  - Doutorandos tinham vários anos de experiência no ramo de sistemas operacionais embarcados.

Validação

- Foi utilizada somente uma SPL, de um domínio específico.
  - SPL é um sistema típico do mercado, ao invés de um software de pesquisa.
  - Resultado pode ser aplicado em outras SPLs, o que é de interesse do mercado.

Conclusão

- Programadores que trabalham com cores são mais rápidos em algumas tarefas.
- Programadores acham as cores mais agradáveis e adequadas para se trabalhar com grandes SPLs.
- Uso de cores para melhorar o entendimento de programas é promissor!