Taking a look inside the Java Virtual Machine

1. System scope

   Het project bestaat uit het ontwikkelen van een profiler die gebruik van de JVMTI interface. Met deze interface kunnen agents geschreven worden die aan de JVM kunnen meegegeven worden om acties te loggen, analyseren, ...

   In de eerste fase moet de profiler in staat zijn om:
   - Een programma selectief te tracen: De method-calls loggen, return calls.
   - De mogelijkheid bieden om object-identifiers mee te loggen.
   - Optioneel de mogelijkheid bieden om de waarden van parameters te loggen.
   - De mogelijkheid bieden om deze gegevens per thread te loggen of algemeen.
   - Alle opties moeten in en uitgeschakeld kunnen worden.
   - Deze gegevens uiteraard achteraf ook terug weer te geven.

   In de tweede fase moet het mogelijk zijn om een momentopname van de JVM weer te geven:
   - Code coverage, cpu verbruik: Percentage/ms doorgebracht in een methode, aantal invocaties, liefst weergegeven in een boomstructuur (per thread/algemeen).
   - Het verloop van de threads, en de staat waarin ze zich bevinden(running, suspended, ...).
   - Gebruik van stack en heap.
   - Hoeveel instanties van een bepaalde klasse er bestaan en het geheugen dat ze innemen.
   - Een filter op deze weergaven toe te passen.
   - De weergaven on-the-fly in en uit te schakelen.

2. Risico analyse
   a. Menselijke risico's
      - Ziekte van projectlid
        Waarschijnlijkheid: Gemiddeld
        Impact: Tolerabel
        Tegenmaatregel: Bufferperiodes bij mijlpalen plannen om de eventueel verloren tijd in te halen.

      - Het tekortschieten van de kennis van de projectleden (mbt. de JVMTI interface).
        Waarschijnlijkheid: Hoog
        Impact: Ernstig
        Tegenmaatregel: De interface op voorhand goed bestuderen en vastleggen welke delen we zeker gaan gebruiken.

      - Te weinig tijd door andere projecten.
        Waarschijnlijkheid: Gemiddeld
        Impact: Ernstig
        Tegenmaatregel: Goede planning. Moest het toch voorvallen dan zullen het nachtjes door worden.

b. Technische risico's

- Het betreft hier een tamelijk nieuwe interface waarover nog niet heel veel documentatie/ervaring te vinden is. Moeilijke interfaces zijn dikwijls makkelijker te begrijpen dmv voorbeelden of tutorials die we moeilijk of niet kunnen vinden.
  Waarschijnlijkheid: Hoog
  Impact: Tolerabel tot serieus
  Tegenmaatregel: Aangezien we waarschijnlijk moeten terugvallen op de specificaties van Sun, deze op voorhand zo goed mogelijk bestuderen om eventuele moeilijkheden op voorhand te kunnen identificeren.

- Bugs in de JVMTI interface
  Waarschijnlijkheid: Onbeduidend
  Impact: Zeer ernstig (louche trucs nodig om deze problemen op te lossen?)
  Tegenmaatregel: Hier kan weinig tegen gedaan worden.

- Platformafhankelijkheden. De agents moeten in C of C++ geschreven worden. Het kan zijn dat we gebruik maken van bepaalde bibliotheken die maar voor een beperkt aantal platformen beschikbaar zijn.
  vb: WinSock, Unix Sockets, ...
  Waarschijnlijkheid: Gemiddeld
  Impact: Serieus
  Tegenmaatregel: De doelplatformen op voorhand bepalen, en vastleggen welke bibliotheken de agents nodig zullen hebben. Eventueel alternatieven zoeken (WinSock vs Unix Sockets) die met preprocessor code geactiveerd kunnen worden.

3. Taken

Fase 1:

1) Nieuw programma tracen

a. Nieuwe sessie starten
   Tijd: 1 week
   Prioriteit: Verplicht
b. Agent
   Tijd: 2 weken
   Prioriteit: Verplicht

Mijlpaal: Een programma kan getraced worden.

2) Log bekijken

a. Programmaverloop bekijken
   Tijd: 1 week
   Prioriteit: Verplicht
   Mijlpaal: De log kan bekeken worden
b. Samenvatting bekijken
   Tijd: 2 weken
   Prioriteit: Nice to have
   Mijlpaal: Er kan een samenvatting van de log gegenereerd worden
c. Samenvatting opslaan
   Tijd: 2 dagen.
   Prioriteit: Nice to have

Mijlpaal: De samenvatting kan opgeslagen worden.

3) **Filters voor log**
   Tijd: 1 week
   Prioriteit: Belangrijk
   Mijlpaal: Er kan op het log gefilterd worden (methodes, threads, ...)
4) **Uitvoering manueel stoppen**
   Tijd: 2 dagen
   Prioriteit: Belangrijk
   Mijlpaal: De uitvoering van het programma kan gestopt worden.
5) **Uitvoering automatisch stoppen wanneer de log te groot wordt.**
   Tijd: 1 week
   Prioriteit: Nice to have
   Mijlpaal: Als de log te groot wordt, zal het programma automatisch afgebroken worden.

Fase 2:

6) **Code coverage**
   Tijd: 4 weken
   Prioriteit: Verplicht
   Mijlpaal: Code-coverage on-the-fly weergeven. De weergave moet ten alle tijd in- en uitgeschakeld kunnen worden.
7) **Threadverloop**
   Tijd: 2 weken
   Prioriteit: Belangrijk
   Mijlpaal: De huidige status van de threads (en directe verleden) on-the-fly weergeven. De weergave moet ten alle tijd in- en uitgeschakeld kunnen worden.
8) **Gebruik van stack en heap**
   Tijd: 3 weken
   Prioriteit: Verplicht
   Mijlpaal: De staat van de stack en heap on-the-fly weergeven. De weergave moet ten alle tijd in- en uitgeschakeld kunnen worden.
9) **Instanties van klassen**
   Tijd: 2 weken
   Prioriteit: Belangrijk
   Mijlpaal: Het aantal instanties van de klassen, en het verbuik van het geheugen on-the-fly weergeven. De weergave moet ten alle tijd in- en uitgeschakeld kunnen worden.
10) ***Afwerken van de applicatie, debuggen.***
   Tijd: 1 week
   Prioriteit: Laag

4. Planning

Maand	Week	Ma	Di	Woe	Do	Vr	Za	Zo
Oktober	1							
	2							
	3							
	4	requirements	Fase 1					
	5							
November	1	1a						
	2	1b						
	3							
	4	2a						
	5	3						
December	1							
	2	4		2b				
	3			2c				
	4			5				
	5							
Januari	1							
	2							
	3							
	4							
	5							
	6	Fase 2						
Februari	1							
	2							
	3	6						
	4							
	5							
Maart	1							
	2							
	3	8						
	4							
	5							
April	1							
	2	9						
	3							
	4	7						
	5							
Mei	1							
	2	10						
	3							
	4							
	5							
	6							
Juni	1							
	2							
	3							
	4							

Legenda	
Buffer	
Design	
Implementatie	
Documentatie	
Negeren	