

Systemes multi-agents

Cours 4 – Architectures cognitives: Belief–Desire–Intention

Cédric Buron

`cedric.buron@yahoo.fr` | `buron.cedric.free.fr`

Ingénieur de recherche décision

THALES



INTRODUCTION

Agents cognitifs

Caractéristiques :

- représentation du monde
- capable de raisonner
- architecture de type sense→plan→act

Architectures communes :

- Belief Desire Intention (BDI)
- agents logiques
- processus markoviens

Agents cognitifs

Caractéristiques :

- représentation du monde
- capable de raisonner
- architecture de type sense→plan→act

Architectures communes :

- Belief Desire Intention (BDI)
- agents logiques
- processus markoviens

Principes de l'architecture BDI

- Basé sur un modèle de la psychologie humaine [1]
- Composé de 4 composants :
 1. Beliefs : croyances de l'agents sur l'état du monde, sur son propre état, sur les autres agents etc.
 2. Desires : buts de l'agent, objectifs généraux de l'agent
 3. Intentions : buts de l'agent à court terme, objectifs qu'il essaie d'atteindre actuellement
 4. Plans : ensemble des plans accessibles à l'agent, composés d'une suite d'actions atomiques.

Quelles contraintes sur ces ensembles ?

Logique modale épistémique

- logique servant à représenter les connaissances/croyances d'un agent
- $K_i\varphi$: « la proposition φ est crue(/désirée/intention) par l'agent i »
- axiomatique sur un ensemble de propositions (ou modèle¹) M :
 - axiome de distribution \mathbf{K}^2 : $(K_i\varphi \wedge K_i(\varphi \implies \psi)) \implies K_i\psi$
 - axiome de cohérence \mathbf{D} : $\neg(K_i\varphi \wedge K_i\neg\varphi)$
 - axiome d'introspection positive $\mathbf{4}$: $K_i\varphi \implies K_i(K_i(\varphi))$
 - axiome d'introspection négative $\mathbf{5}$: $\neg K_i\varphi \implies K_i(\neg K_i(\varphi))$
- Modèles \mathcal{B} , \mathcal{D} et \mathcal{I} :
 - ▶ \mathcal{B} : KD45
 - ▶ \mathcal{D} : KD
 - ▶ \mathcal{I} : KD

1. mauvaise traduction de *modal*

2. \wedge : et logique; \implies implication logique; *neg* : non logique

Logique modale épistémique

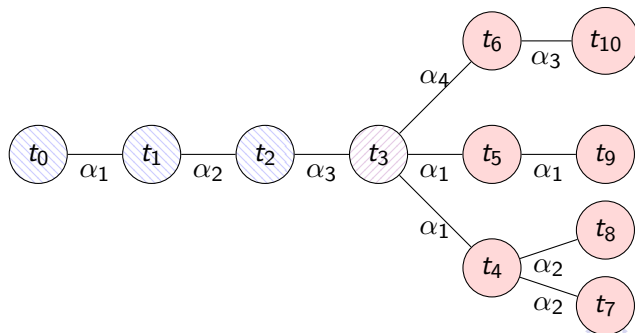
Attention aux confusions !

- « Je désire avoir une bonne note à ce module et avoir une bonne note à ce module implique de travailler donc je désire travailler »
- « Je désire que mes étudiants aient une bonne note et je désire qu'avoir une bonne note implique que l'étudiant a bien travaillé donc je désire que mes étudiants travaillent bien. »
- « Je désire avoir une bonne note à ce module et avoir une bonne note à ce module implique de finir le TP ce soir. Je désire aussi aller au concert de Vadim Repin, et aller à ce concert implique de ne pas finir mon TP ce soir. »

Logic Of Rational Agents (Wooldridge)

Composition de plusieurs types de logiques :

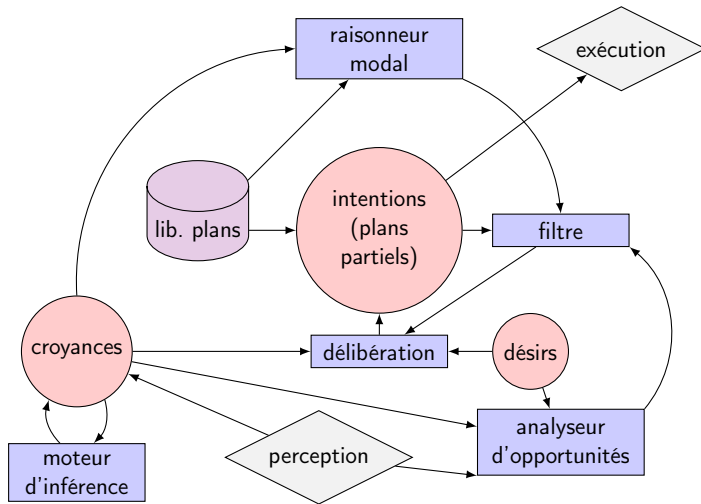
- logique modale
- logique temporelle (ou dynamique) :
 - ▶ temps représenté comme un arbre
 - ▶ passé linéaire
 - ▶ futur représenté par des branchements de possibilité
 - ▶ transitions du monde annotées par des actions



ARCHITECTURES

BDI

IRMA (Bratman et al.)



IRMA (Bratman *et al.*)

Composants

révision de croyances met à jour la base de croyances de l'agent en fonction des perceptions reçues. S'assure que la base de croyance reste KD45

délibération génère un ensemble d'intentions et de désirs cohérents à partir des croyances, des désirs et des intentions. S'assure des propriétés (resp. K et KD) de ces ensembles

analyseur d'opportunités détermine les intentions les plus à même d'aboutir, et celles qui n'aboutiront pas

raisonneur modal transforme des intentions en plans partiels

filtre utilise les désirs et les intentions et s'appuie sur l'analyseur d'opportunités et le raisonneur modal pour générer des plans partiels (plans de haut niveau)

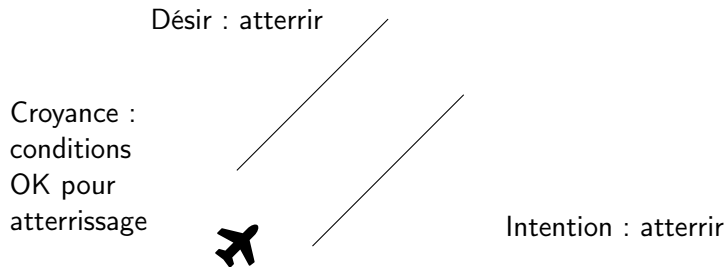
librairie de plans librairie de plans préétablis ; permet de transformer les plans partiels en plans exécutables.

IRMA (Bratman *et al.*)

Intelligent Resource-bounded Machine Architecture

- Première architecture BDI
- Pas implémentée
- Basée sur des composants et des interconnexions

Exemple :

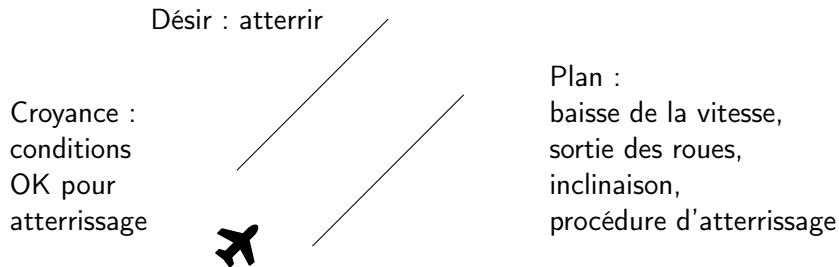


IRMA (Bratman *et al.*)

Intelligent Resource-bounded Machine Architecture

- Première architecture BDI
- Pas implémentée
- Basée sur des composants et des interconnexions

Exemple :

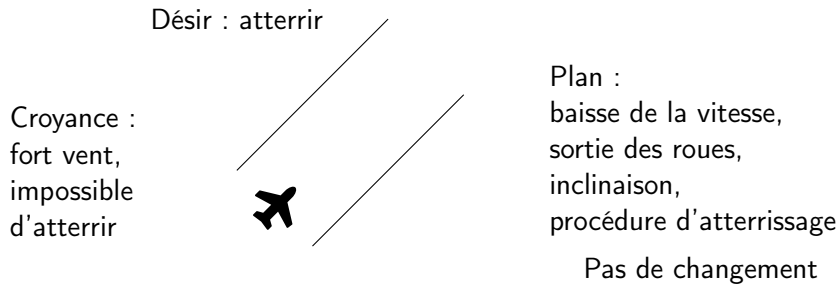


IRMA (Bratman *et al.*)

Intelligent Resource-bounded Machine Architecture

- Première architecture BDI
- Pas implémentée
- Basée sur des composants et des interconnexions

Exemple :



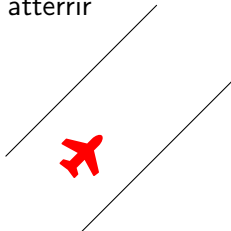
IRMA (Bratman *et al.*)

Intelligent Resource-bounded Machine Architecture

- Première architecture BDI
- Pas implémentée
- Basée sur des composants et des interconnexions

Exemple :

Désir : atterrir



Plan :
baisse de la vitesse,
sortie des roues,
inclinaison,
procédure d'atterrissage

Engagement

Quel problème ? Quelles solutions ?

Engagement

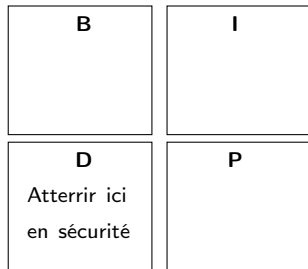
Quel problème ? Quelles solutions ?

- problème : l'agent est « obstiné »
- solution : révision des plans/intentions selon la notion d'*engagement*
- 3 niveaux d'engagement :
 - engagement aveugle* (*blind commitment*) révision lorsque l'intention est atteinte
 - engagement obstiné* (*single-minded commitment*) idem + remise en cause des intentions si l'agent ne les croit plus réalisables
 - engagement ouvert* (*open-minded commitment*) idem + remise en cause des intentions lorsque les désirs ont changé

Engagement : la boucle BDI (Wooldridge)

Engagement aveugle

```
 $B \leftarrow B_0; D \leftarrow D_0; I \leftarrow I_0$   
loop  
  Obtenir nles perceptions  $p$ ●  
   $B \leftarrow \text{recv}(B, p)$   
   $I \leftarrow \text{options}(D, I)$   
   $D \leftarrow \text{des}(B, D, I)$   
   $I \leftarrow \text{filtre}(B, D, I)$   
   $PE \leftarrow \text{plan}(B, I)$   
  Exécuter(PE)  
end loop
```



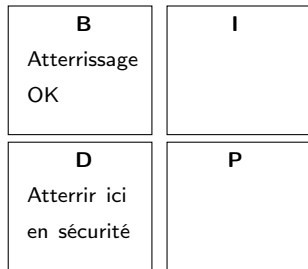
« Att.
OK » ;
Temps clair
Pas de vent



Engagement : la boucle BDI (Wooldridge)

Engagement aveugle

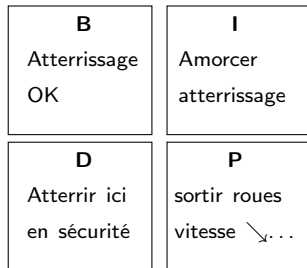
```
 $B \leftarrow B_0; D \leftarrow D_0; I \leftarrow I_0$   
loop  
  Obtenir nles perceptions  $p$   
   $B \leftarrow \text{recv}(B, p)$   
   $I \leftarrow \text{options}(D, I)$   
   $D \leftarrow \text{des}(B, D, I)$   
   $I \leftarrow \text{filtre}(B, D, I)$   
   $PE \leftarrow \text{plan}(B, I)$   
  Exécuter(PE)  
end loop
```



Engagement : la boucle BDI (Wooldridge)

Engagement aveugle

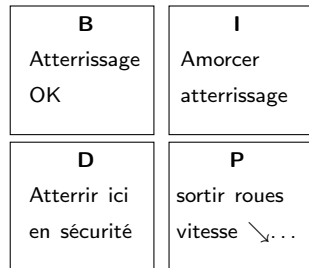
```
 $B \leftarrow B_0; D \leftarrow D_0; I \leftarrow I_0$   
loop  
  Obtenir nles perceptions  $p$   
   $B \leftarrow \text{recv}(B, p)$   
   $I \leftarrow \text{options}(D, I) \bullet$   
   $D \leftarrow \text{des}(B, D, I) \bullet$   
   $I \leftarrow \text{filtre}(B, D, I) \bullet$   
   $PE \leftarrow \text{plan}(B, I) \bullet$   
  Exécuter(PE)  
end loop
```



Engagement : la boucle BDI (Wooldridge)

Engagement aveugle

```
 $B \leftarrow B_0; D \leftarrow D_0; I \leftarrow I_0$   
loop  
  Obtenir nles perceptions  $p$   
   $B \leftarrow \text{recv}(B, p)$   
   $I \leftarrow \text{options}(D, I)$   
   $D \leftarrow \text{des}(B, D, I)$   
   $I \leftarrow \text{filtre}(B, D, I)$   
   $PE \leftarrow \text{plan}(B, I)$   
  Exécuter( $PE$ )•  
end loop
```



« Att.
ailleurs »



Engagement : la boucle BDI (Wooldridge)

```
1:  $B \leftarrow B_0; D \leftarrow D_0; I \leftarrow I_0$ 
2: loop
3:   Obtenir nlls perceptions  $p$ 
4:    $B \leftarrow \text{recv}(B, p)$ 
5:    $I \leftarrow \text{options}(D, I)$ 
6:    $D \leftarrow \text{des}(B, D, I)$ 
7:    $I \leftarrow \text{filtre}(B, D, I)$ 
8:    $PE \leftarrow \text{plan}(B, I)$ 
9:   Exécuter(PE)
10: end loop
```

Engagement : la boucle BDI (Wooldridge)

Engagement obstiné

$B \leftarrow B_0; D \leftarrow D_0; I \leftarrow I_0$

loop

Obtenir nlls perceptions p

$B \leftarrow \text{recv}(B, p)$

$I \leftarrow \text{options}(D, I)$

$D \leftarrow \text{des}(B, D, I)$

$I \leftarrow \text{filtre}(B, D, I)$

$PE \leftarrow \text{plan}(B, I)$

while ($PE \neq \emptyset \wedge \text{possible}(I, B)$) **do**

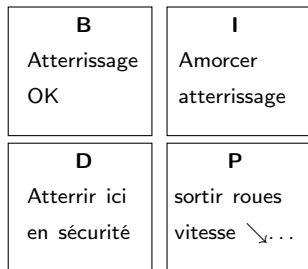
$x \leftarrow PE.\text{pop}(); \text{exec}(x)$

obtenir nlls perceptions p

$B \leftarrow \text{recv}(B, p)$

end while

end loop



« Att.
ailleurs »



Engagement : la boucle BDI (Wooldridge)

Engagement obstiné

$B \leftarrow B_0; D \leftarrow D_0; I \leftarrow I_0$

loop

Obtenir nlls perceptions p

$B \leftarrow \text{recv}(B, p)$

$I \leftarrow \text{options}(D, I)$

$D \leftarrow \text{des}(B, D, I)$

$I \leftarrow \text{filtre}(B, D, I)$

$PE \leftarrow \text{plan}(B, I)$

while ($PE \neq \emptyset \wedge \text{possible}(I, B)$) **do**

$x \leftarrow PE.\text{pop}(); \text{exec}(x)$ •

obtenir nlls perceptions p •

$B \leftarrow \text{revc}(B, p)$ •

end while

end loop

B Atterrissage OK/ailleurs?	I Amorcer atterrissage
D Atterrir ici en sécurité	P sortir roues vitesse ↘...



Engagement : la boucle BDI (Wooldridge)

Engagement obstiné

$B \leftarrow B_0; D \leftarrow D_0; I \leftarrow I_0$

loop•

Obtenir nlls perceptions p

$B \leftarrow \text{recv}(B, p)$

$I \leftarrow \text{options}(D, I)$

$D \leftarrow \text{des}(B, D, I)$

$I \leftarrow \text{filtre}(B, D, I)$

$PE \leftarrow \text{plan}(B, I)$

while ($PE \neq \emptyset \wedge \text{possible}(I, B)$)• **do**

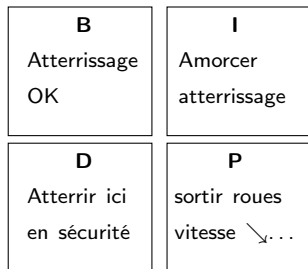
$x \leftarrow PE.\text{pop}(); \text{exec}(x)$

obtenir nlls perceptions p

$B \leftarrow \text{recv}(B, p)$

end while

end loop



Fort vent
orage



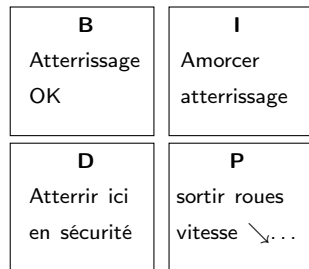
Engagement : la boucle BDI (Wooldridge)

```
1:  $B \leftarrow B_0; D \leftarrow D_0; I \leftarrow I_0$ 
2: loop
3:   Obtenir nlls perceptions  $p$ 
4:    $B \leftarrow \text{recv}(B, p)$ 
5:    $I \leftarrow \text{options}(D, I)$ 
6:    $D \leftarrow \text{des}(B, D, I)$ 
7:    $I \leftarrow \text{filtre}(B, D, I)$ 
8:    $PE \leftarrow \text{plan}(B, I)$ 
9:   while ( $PE \neq \emptyset \wedge \text{possible}(I, B)$ ) do
10:     $x \leftarrow PE.\text{pop}(); \text{exec}(x)$ 
11:    obtenir nlls perceptions  $p$ 
12:     $B \leftarrow \text{recv}(B, p)$ 
13:  end while
14: end loop
```

Engagement : la boucle BDI (Wooldridge)

Engagement ouvert

```
while ( $PE \neq \emptyset \wedge possible(I, B)$ ) do●  
   $x \leftarrow PE.pop()$ ;  $exec(x)$   
  obtenir nlls perceptions  $p$   
   $B \leftarrow revc(B, p)$   
   $D \leftarrow des(B, D, I)$   
   $I \leftarrow filtre(B, D, I)$   
   $PE \leftarrow plan(B, I)$   
end while
```



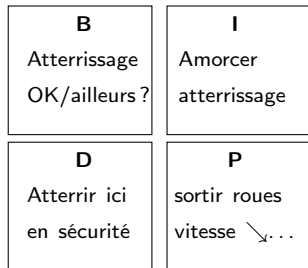
« Att.
ailleurs »



Engagement : la boucle BDI (Wooldridge)

Engagement ouvert

```
while ( $PE \neq \emptyset \wedge possible(I, B)$ ) do
   $x \leftarrow PE.pop()$ ;  $exec(x)$ •
  obtenir nlls perceptions  $p$ •
   $B \leftarrow revc(B, p)$ •
   $D \leftarrow des(B, D, I)$ 
   $I \leftarrow filtre(B, D, I)$ 
   $PE \leftarrow plan(B, I)$ 
end while
```



Engagement : la boucle BDI (Wooldridge)

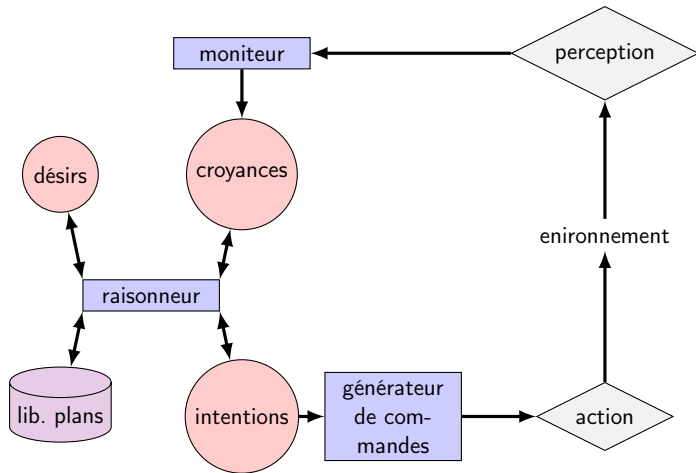
Engagement ouvert

```
while ( $PE \neq \emptyset \wedge possible(I, B)$ ) do  
   $x \leftarrow PE.pop()$ ;  $exec(x)$   
  obtenir nlls perceptions  $p$   
   $B \leftarrow revc(B, p)$   
   $D \leftarrow des(B, D, I)$ •  
   $I \leftarrow filtre(B, D, I)$ •  
   $PE \leftarrow plan(B, I)$ •  
end while
```

B Atterrissage OK	I Trouver aéroport
D Atterrir ailleurs	P Com. autres aéroports



Engagement : Procedural Reasoning System (Georgeff & Lansky)



Langages orientés BDI I

AgentSpeak langage de programmation *abstrait* basé sur PRS

dMARS Distributed Multi-Agent Reasoning System

- langage de développement BDI basé sur PRS,
- Développé en C++,
- Plus supporté aujourd'hui (*cf.* JACK).

JACK plate-forme propriétaire de Agent Oriented Software Ltd. Caractéristiques :

- inspiré de PRS et dMars
- Domain Specific Language (DSL) compilé en Java
- IDE fourni (facilités de conception)
- noyau avec fonctionnalités de base (messages, raisonnement...)
- Extensions : FIPA, déploiement sur serveur...

Jason implémentation Java de AgentSpeak

- opensource (GPL/LGPL)
- extension de AgentSpeak plus pratique

Langages orientés BDI II

- interconnexion possible avec d'autres frameworks (JADE)

Jadex Extension de JADE implémentant le modèle BDI

- développement Java + XML
- embarque une interface et un IDE
- opensource LGPL
- inclut toutes les fonctionnalités de JADE

2APL A Practical Agent Programming Language

- Domain Specific Language + Java
- IDE (plugin Eclipse)
- opensource (GPL 3)

GOAL langage orienté but







- Domain Specific Language (semblable à PROLOG)
- IDE (plugin Eclipse)
- opensource

Conclusion

- Avantages de l'approche BDI
 - ▶ pratique : agent capable de raisonner
 - ▶ « proche de l'humain » (basé sur une théorie de la cognition humaine)
 - ▶ explicable, prédictible
 - ▶ basé sur une axiomatique riche
 - ▶
- Inconvénients
 - ▶ pas de notion d'apprentissage
 - ▶ beaucoup de calibration à la main
 - ▶ beaucoup de calcul

```
git clone https://gitlab.data-ensta.fr/buron/2020-2021-ia310-cours-4.git  
http://cedricburon.eu/cours/TP4.zip
```




Références I

-  Michael E Bratman. *Intention, Plans, and Practical Reason*, Harvard University Press, 1987.
-  Michael Wooldridge. *Reasoning about rational agents* MIT Press, 2000.
-  Rafael H Bordini, et Jomi F Hübner. “BDI agent programming in AgentSpeak using Jason.” *International Workshop on Computational Logic in Multi-Agent Systems*, pp. 143-164, 2005.
-  Michael E Bratman, David J Israel et Martha E Pollack. “Plans and resource-bounded practical reasoning”. *Computational Intelligence* 4.4 pp 349-355, 1988.
-  Mehdi Dastani. “2APL : a practical agent programming language.” *Autonomous agents and multi-agent systems* 16.3, pp 214-248, 2008.
-  Michel P Georgeff, Amy L Lansky. “Procedural knowledge”. *Proceedings of the IEEE* 74.10 pp 1383-1398, 1986.

Références II

-  Michel P Georgeff et François-Félix Ingrand. "Decision making in an embedded reasoning system". *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* pp 972-978, 1989.
-  Koen V. Hindriks "Programming rational agents in goal." *Multi-Agent Programming*, pp. 119-157, 2009.
-  Nick Howden, Ralph Rönquist, Andrew Hodgson, et Andrew Lucas. "JACK intelligent agents-summary of an agent infrastructure." *5th International conference on autonomous agents*, 2001.
-  Mark d'Inverno, David Kinny, Michael Luck et Michael Wooldridge. "A formal specification of dMARS." *International Workshop on Agent Theories, Architectures, and Languages*, pp 155-176, 1997.
-  Alexander Pokahr, Lars Braubach, et Winfried Lamersdorf. "Jadex : A BDI reasoning engine." *Multi-agent programming* pp 149-174, 2005.

Références III

-  Anand S Rao et Michael P Georgeff. "BDI agents : from theory to practice". *Proceedings of the first international conference on multiagent systems* pp 312-319, 1995.
-  Anand S Rao "AgentSpeak (L) : BDI agents speak out in a logical computable language." *European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, pp 42-55, 1996.
-  M S Smitha Rao, A M Jyothsna. "BDI : Applications and Architectures". *IJERT 2013 2.2*, 2013.