

# Discuter:

Ontologies + NLP + Minecraft = 🤔

Rafael Accácio NOGUEIRA

`rafael.accacio.nogueira@gmail.com`

29 août 2024 @ Toulouse

# DISCUTER

Dialogue Interactif Structuré, Consolidé et Unifié pour la réalisation de Tâches En Robotique

# DISCUTER

Dialogue Interactif Structuré, Consolidé et Unifié pour la réalisation de Tâches En Robotique

Étudier la collaboration entre agents pour des tâches complexes en utilisant :

# DISCUTER

Dialogue Interactif Structuré, Consolidé et Unifié pour la réalisation de Tâches En Robotique

Étudier la collaboration entre agents pour des tâches complexes en utilisant :

- ▶ langage naturel

# DISCUTER

Dialogue Interactif Structuré, Consolidé et Unifié pour la réalisation de Tâches En Robotique

Étudier la collaboration entre agents pour des tâches complexes en utilisant :

- ▶ langage naturel
- ▶ événements non-linguistiques

# DISCUTER

Dialogue Interactif Structuré, Consolidé et Unifié pour la réalisation de Tâches En Robotique

Étudier la collaboration entre agents pour des tâches complexes en utilisant :

- ▶ langage naturel
- ▶ événements non-linguistiques
- ▶ contenu des bases de connaissances

# DISCUTER

Dialogue Interactif Structuré, Consolidé et Unifié pour la réalisation de Tâches En Robotique

Étudier la collaboration entre agents pour des tâches complexes en utilisant :

- ▶ langage naturel
- ▶ événements non-linguistiques
- ▶ contenu des bases de connaissances
- ▶ états cognitifs modélisés

# DISCUTER

Dialogue Interactif Structuré, Consolidé et Unifié pour la réalisation de Tâches En Robotique

Étudier la collaboration entre agents pour des tâches complexes en utilisant :

- ▶ langage naturel
- ▶ événements non-linguistiques
- ▶ contenu des bases de connaissances
- ▶ états cognitifs modélisés

Mais quelle activité/outils pour tester ?

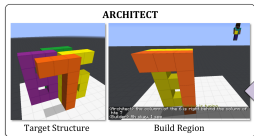


# Collaborative Dialogue

NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER, « Collaborative Dialogue in Minecraft », 2019

# Collaborative Dialogue

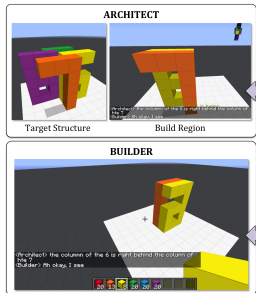
NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER, « Collaborative Dialogue in Minecraft », 2019



Source : NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER 2019

# Collaborative Dialogue

NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER, « Collaborative Dialogue in Minecraft », 2019

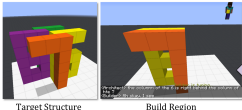


Source : NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER 2019

# Collaborative Dialogue

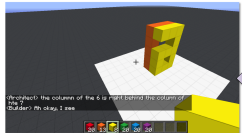
NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER, « Collaborative Dialogue in Minecraft », 2019

**ARCHITECT**



Target Structure      Build Region

**BUILDER**



Builder's chat interface showing Architect's instructions:

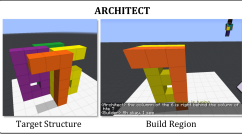
Architect: in about the middle build a column five tall  
(Builder puts down five orange blocks)  
Architect: then two more to the left of the top to make a 7  
(Builder puts down two orange blocks)  
Architect: now a yellow 6  
Architect: the long edge of the 6 aligns with the stem of the 7 and faces right  
Builder: Where does the 6 start?  
Architect: behind the 7 from your perspective  
Builder: Is it directly adjacent?  
Architect: yes directly behind it. touches it  
(Builder puts down twelve yellow blocks, in the shape of a 6)  
Architect: too much overlap unfortunately  
Architect: the column of the 6 is right behind the column of the 7

Source : NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER 2019

# Collaborative Dialogue


NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER, « Collaborative Dialogue in Minecraft », 2019

**ARCHITECT**



Target Structure      Build Region

**BUILDER**



Builder's view showing the yellow 6x6x6 cube placed behind the orange structure.

**CHAT INTERFACE**

**Architect:** in about the middle build a column five tall  
*(Builder puts down five orange blocks)*

**Architect:** then two more to the left of the top to make a 7  
*(Builder puts down two orange blocks)*

**Architect:** now a yellow 6

**Architect:** the long edge of the 6 aligns with the stem of the 7 and faces right

**Builder:** Where does the 6 start?

**Architect:** behind the 7 from your perspective

**Builder:** Is it directly adjacent?

**Architect:** yes directly behind it. touches it  
*(Builder puts down twelve yellow blocks, in the shape of a 6)*

**Architect:** too much overlap unfortunately

**Architect:** the column of the 6 is right behind the column of the 7

Source : NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER 2019

Collection des Dialogues<sup>1</sup>



# Collaborative Dialogue

NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER, « Collaborative Dialogue in Minecraft », 2019

The screenshot displays the Minecraft collaborative dialogue interface. It is divided into three main sections: **ARCHITECT**, **BUILDER**, and **CHAT INTERFACE**.

**ARCHITECT** section contains two sub-images: **Target Structure** (a 3D model of a structure made of purple, orange, and yellow blocks) and **Build Region** (a 3D model of the same structure on a flat white ground).

**BUILDER** section shows a 3D model of the structure being built on a flat white ground, with a small yellow block being placed. A chat log is visible at the bottom of the Builder window.

**CHAT INTERFACE** section contains a list of messages:

- Architect:** in about the middle build a column five tall  
*(Builder puts down five orange blocks)*
- Architect:** then two more to the left of the top to make a 7  
*(Builder puts down two orange blocks)*
- Architect:** now a yellow 6
- Architect:** the long edge of the 6 aligns with the stem of the 7 and faces right
- Builder:** Where does the 6 start?
- Architect:** behind the 7 from your perspective
- Builder:** Is it directly adjacent?
- Architect:** yes directly behind it. touches it  
*(Builder puts down twelve yellow blocks, in the shape of a 6)*
- Architect:** too much overlap unfortunately
- Architect:** the column of the 6 is right behind the column of the 7


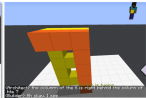
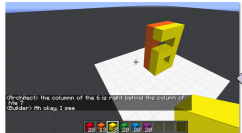
Source : NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER 2019

Collection des Dialogues<sup>1</sup> → Génération d'énonciations



# Collaborative Dialogue

NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER, « Collaborative Dialogue in Minecraft », 2019

ARCHITECT		CHAT INTERFACE
		
Target Structure		<p><b>Architect:</b> in about the middle build a column five tall <i>(Builder puts down five orange blocks)</i></p> <p><b>Architect:</b> then two more to the left of the top to make a 7 <i>(Builder puts down two orange blocks)</i></p> <p><b>Architect:</b> now a yellow 6</p> <p><b>Architect:</b> the long edge of the 6 aligns with the stem of the 7 and faces right</p> <p><b>Builder:</b> Where does the 6 start?</p> <p><b>Architect:</b> behind the 7 from your perspective</p> <p><b>Builder:</b> Is it directly adjacent?</p> <p><b>Architect:</b> yes directly behind it. touches it <i>(Builder puts down twelve yellow blocks, in the shape of a 6)</i></p> <p><b>Architect:</b> too much overlap unfortunately</p> <p><b>Architect:</b> the column of the 6 is right behind the column of the 7</p>
Build Region		
Builder		
		
Builder		

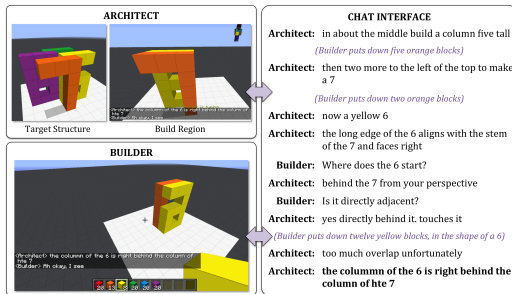
Source : NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER 2019

Collection des Dialogues<sup>1</sup> → Génération d'énonciations (Architect est un robot)



# Collaborative Dialogue

NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER, « Collaborative Dialogue in Minecraft », 2019



Source : NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER 2019

Collection des Dialogues<sup>1</sup> → Génération d'énonciations (Architect est un robot)

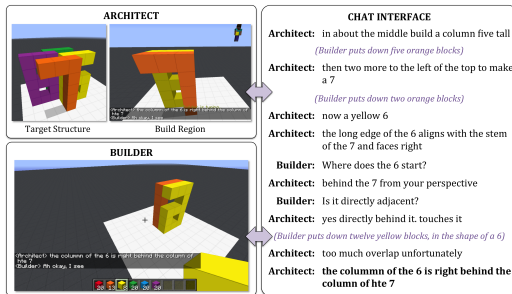
Et si le Builder était un robot ?





# Collaborative Dialogue

NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER, « Collaborative Dialogue in Minecraft », 2019



Source : NARAYAN-CHEN, JAYANNAVAR et HOCKENMAIER 2019

Collection des Dialogues<sup>1</sup> → Génération d'énonciations (Architect est un robot)

Et si le Builder était un robot ?

JAYANNAVAR, NARAYAN-CHEN et HOCKENMAIER 2020 (Pas de raisonnement spatial)

# Travail

Établir l'architecture pour créer une preuve de concept

# Travail

Établir l'architecture pour créer une preuve de concept qui associe

- ▶ Traitement automatique des langues (NLP)
- ▶ Overworld/Ontologenius (Raisonnement spatial et bases de connaissances)

# Travail

Établir l'architecture pour créer une preuve de concept qui associe

- ▶ Traitement automatique des langues (NLP)
- ▶ Overworld/Ontologenius (Raisonnement spatial et bases de connaissances)

## 1. Reliant Minecraft/Overworld/NLP

## 2. Tâches réalisées

## 3. Petite démo

## 4. Conclusion

# Sommaire

## 1. Reliant Minecraft/Overworld/NLP

- A. Architecture
- B. Flux de données
- C. Tâches
- D. Planificateur

# Architecture



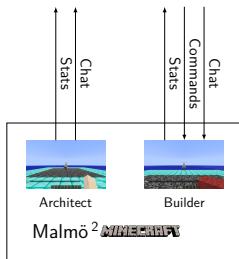
Architect



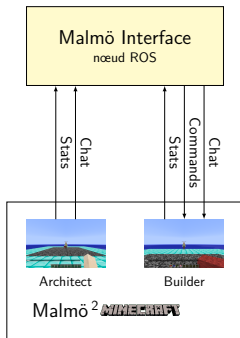
Builder

**MINECRAFT**

# Architecture

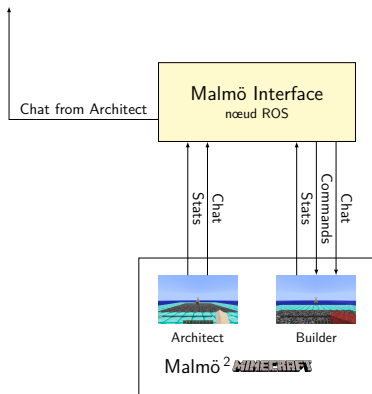


# Architecture

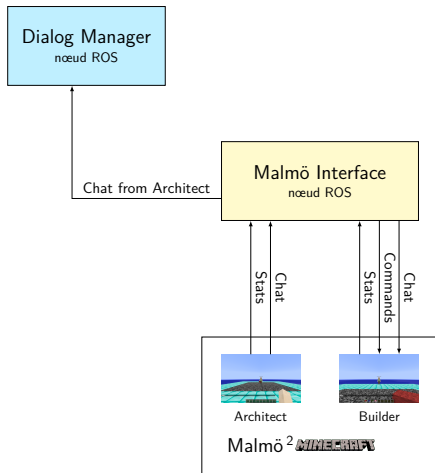




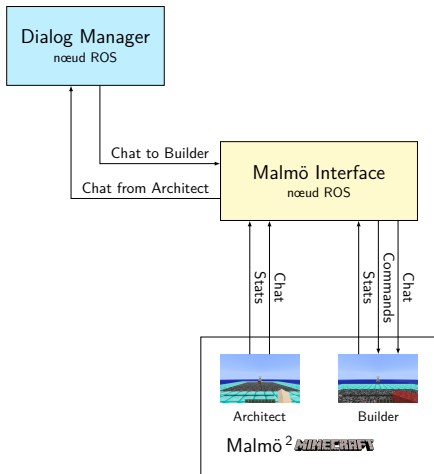
# Architecture



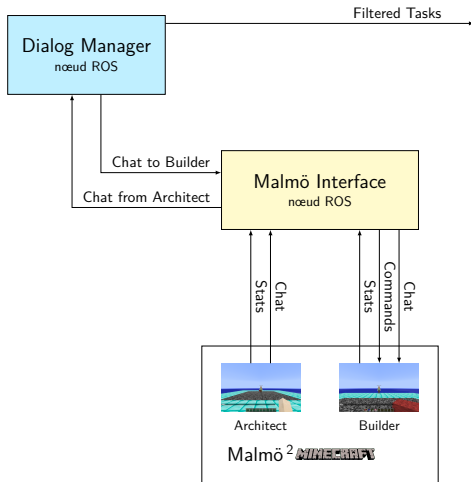
# Architecture



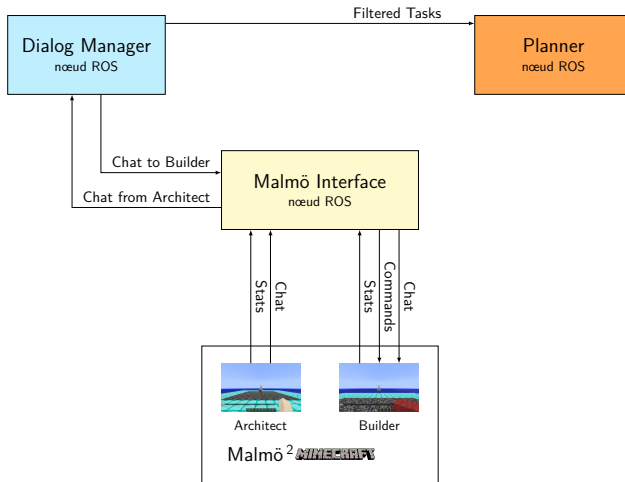
# Architecture



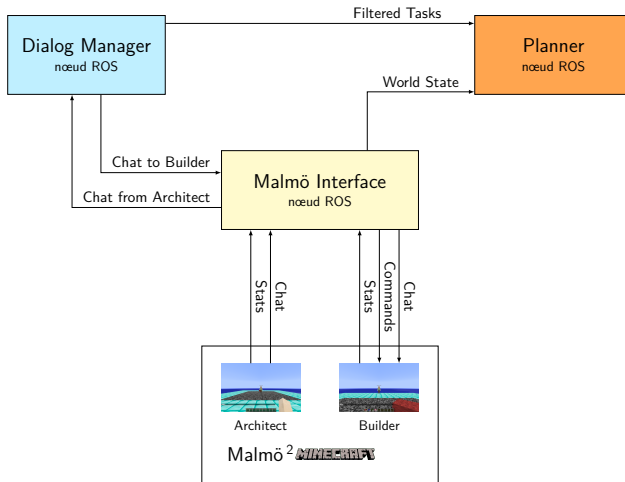
# Architecture



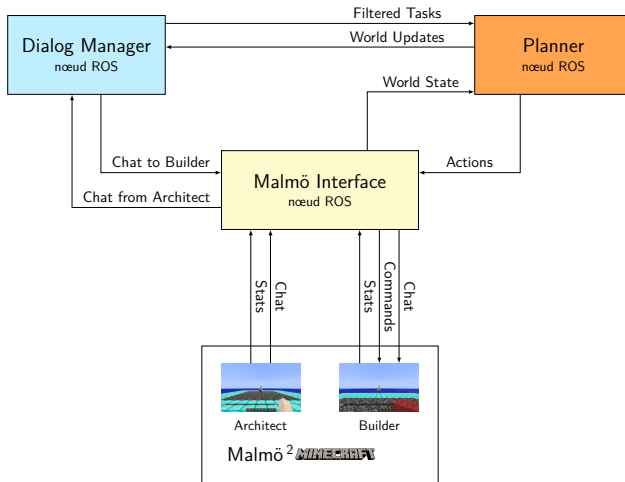
# Architecture



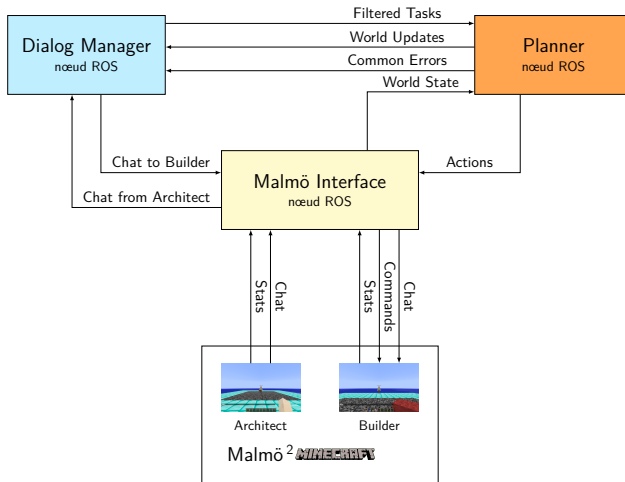
# Architecture



# Architecture



# Architecture





# Flux de données

- ▶ Malmö API
- ▶ ROS
- ▶ Ontologenius

# Flux de données

- ▶ Malmö API (json)
  - ▶ Stats
  - ▶ Chat
  - ▶ Grid Observations
  - ▶ Commands
- ▶ ROS
- ▶ Ontologenius

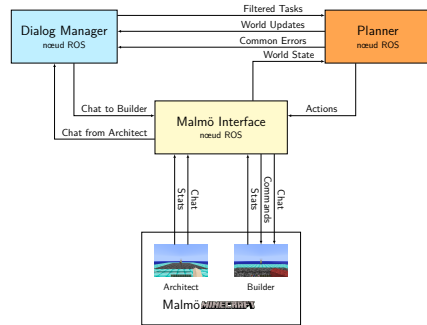
# Flux de données

- ▶ Malmö API (json)
  - ▶ Stats
  - ▶ Chat
  - ▶ Grid Observations
  - ▶ Commands
- ▶ ROS
  - ▶ Chat (StampedString)
  - ▶ (Filtered) Tasks (StampedString)
  - ▶ Common Errors (StampedString)
  - ▶ World Updates (StampedString)
  - ▶ Positions agents/blocs → Overworld (\*Poses)
- ▶ Ontologenius

# Flux de données

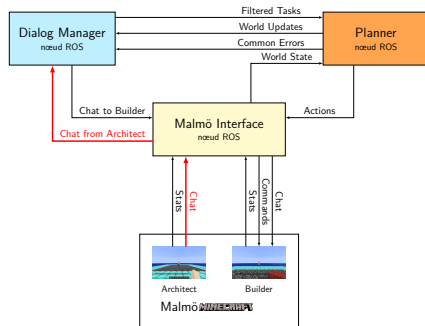
- ▶ Malmö API (json)
  - ▶ Stats
  - ▶ Chat
  - ▶ Grid Observations
  - ▶ Commands
- ▶ ROS
  - ▶ Chat (StampedString)
  - ▶ (Filtered) Tasks (StampedString)
  - ▶ Common Errors (StampedString)
  - ▶ World Updates (StampedString)
  - ▶ Positions agents/blocs → Overworld (\*Poses)
- ▶ Ontologienius
  - ▶ Positions des agents/blocs } Inheritance
  - ▶ Couleurs des blocs } ObjectProperty

# Tâches



# Tâches

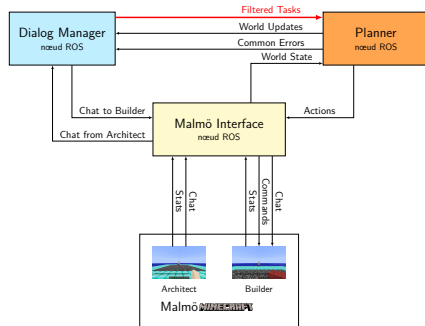
Architect : “Put a blue block on your left”



# Tâches

Architect : “Put a blue block on your left”

Filtered Task possible tâche complexe (PUT(left,blue))

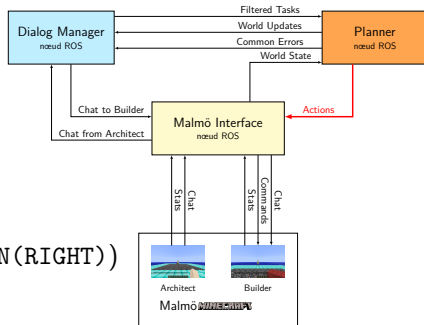


# Tâches

Architect : “Put a blue block on your left”

**Filtered Task** possible tâche complexe (PUT(left,blue))

**Task** liste de tâches (TURN(LEFT), PUT(blue), TURN(RIGHT))





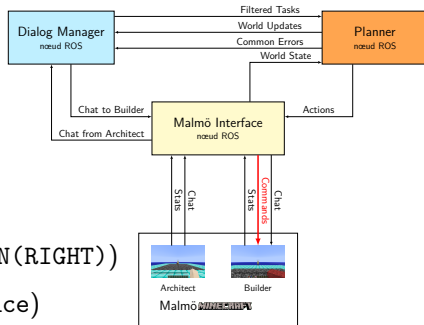
# Tâches

Architect : “Put a blue block on your left”

**Filtered Task** possible tâche complexe (PUT(left,blue))

**Task** liste de tâches (TURN(LEFT), PUT(blue), TURN(RIGHT))

**Commands** Action faite par Builder (turn, setPitch, place)



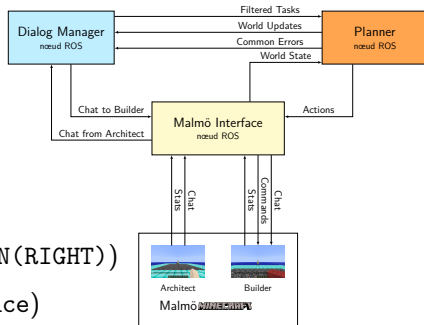
# Tâches

Architect : “Put a blue block on your left”

**Filtered Task** possible tâche complexe (PUT(left,blue))

→ **Task** liste de tâches (TURN(LEFT), PUT(blue), TURN(RIGHT))

**Commands** Action faite par Builder (turn, setPitch, place)



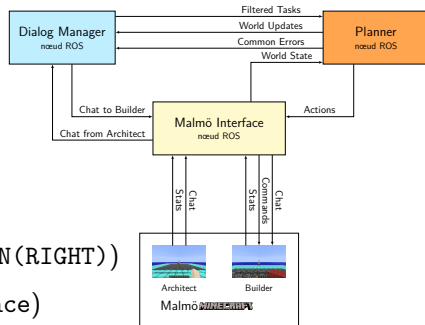
# Tâches

Architect : “Put a blue block on your left”

**Filtered Task** possible tâche complexe (PUT(left,blue))

→ **Task** liste de tâches (TURN(LEFT), PUT(blue), TURN(RIGHT))

**Commands** Action faite par Builder (turn, setPitch, place)



Les tâches doivent

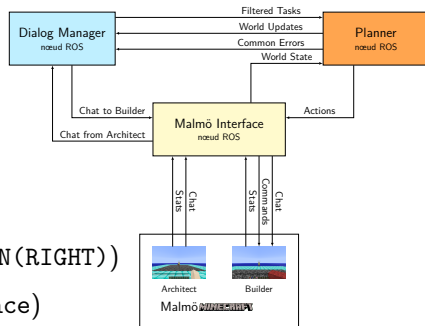
# Tâches

Architect : “Put a blue block on your left”

**Filtered Task** possible tâche complexe (PUT(left,blue))

→ **Task** liste de tâches (TURN(LEFT), PUT(blue), TURN(RIGHT))

**Commands** Action faite par Builder (turn, setPitch, place)



Les tâches doivent

- ▶ être réactives (c-à-d, elles dépendent de l'état)

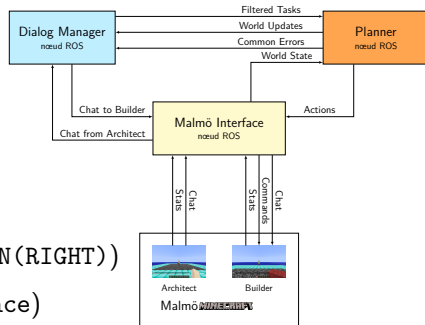
# Tâches

Architect : “Put a blue block on your left”

**Filtered Task** possible tâche complexe (PUT(left,blue))

→ **Task** liste de tâches (TURN(LEFT), PUT(blue), TURN(RIGHT))

**Commands** Action faite par Builder (turn, setPitch, place)



Les tâches doivent

- ▶ être réactives (c-à-d, elles dépendent de l'état)
- ▶ avoir liste de pré-conditions

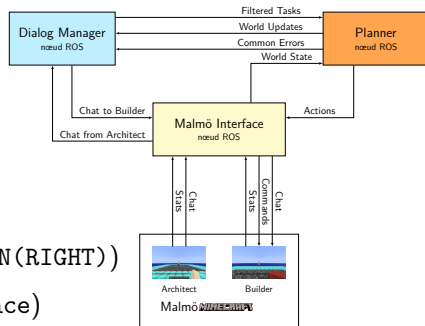
# Tâches

Architect : “Put a blue block on your left”

**Filtered Task** possible tâche complexe (PUT(left,blue))

→ **Task** liste de tâches (TURN(LEFT), PUT(blue), TURN(RIGHT))

**Commands** Action faite par Builder (turn, setPitch, place)



Les tâches doivent

- ▶ être réactives (c-à-d, elles dépendent de l'état)
- ▶ avoir liste de pré-conditions
- ▶ avoir liste de post-conditions

# Planificateur

Partielment implementé

# Planificateur

Partiellement implementé

Planificateur doit :



# Planificateur

Partiellement implementé

Planificateur doit :

avoir une bibliothèque de tâches

# Planificateur

Partiellement implementé

Planificateur doit :

- avoir une bibliothèque de tâches

- avoir accès au raisonnement

# Planificateur

Partiellement implementé

Planificateur doit :

- avoir une bibliothèque de tâches

- avoir accès au raisonnement

- avoir une bibliothèque de conditions et tâches associées (pré et post tâches)

# Planificateur

Partiellement implementé

Planificateur doit :

- avoir une bibliothèque de tâches

- avoir accès au raisonnement

- avoir une bibliothèque de conditions et tâches associées (pré et post tâches)

- tester si action est faisable

# Planificateur

Partiellement implémenté

Planificateur doit :

- avoir une bibliothèque de tâches

- avoir accès au raisonnement

- avoir une bibliothèque de conditions et tâches associées (pré et post tâches)

- tester si action est faisable

- envoyer des m-à-j des actions au Dialog Manager

# Planificateur

Partiellement implementé

Planificateur doit :

- avoir une bibliothèque de tâches

- avoir accès au raisonnement

- avoir une bibliothèque de conditions et tâches associées (pré et post tâches)

- tester si action est faisable

- envoyer des m-à-j des actions au Dialog Manager

- envoyer erreur au Dialog Manager sinon

# Planificateur

Partiellement implémenté

Planificateur doit :

- ✓ avoir une bibliothèque de tâches
- avoir accès au raisonnement
  - avoir une bibliothèque de conditions et tâches associées (pré et post tâches)
  - tester si action est faisable
- ✓ envoyer des m-à-j des actions au Dialog Manager
- ✓ envoyer erreur au Dialog Manager sinon

# Sommaire

## 2. Tâches réalisées



# Tâches réalisées

# Tâches réalisées

Point de départ : Stage Ismail (Enregistrement d'états en .json)

# Tâches réalisées

Point de départ : Stage Ismail (Enregistrement d'états en .json)

- ▶ MàJ Malmö pour fonctionner sur Ubuntu 20.04

# Tâches réalisées

Point de départ : Stage Ismail (Enregistrement d'états en .json)

- ▶ MàJ Malmö pour fonctionner sur Ubuntu 20.04
- ▶ Image docker

# Tâches réalisées

Point de départ : Stage Ismail (Enregistrement d'états en .json)

- ▶ MàJ Malmö pour fonctionner sur Ubuntu 20.04
- ▶ Image docker
- ▶ Reproduction scénario de collecte (Malmö pur)

# Tâches réalisées

Point de départ : Stage Ismail (Enregistrement d'états en .json)

- ▶ MàJ Malmö pour fonctionner sur Ubuntu 20.04
- ▶ Image docker
- ▶ Reproduction scénario de collecte (Malmö pur)
- ▶ Création de l'interface Malmö/ROS/Overworld

# Tâches réalisées

Point de départ : Stage Ismail (Enregistrement d'états en .json)

- ▶ MàJ Malmö pour fonctionner sur Ubuntu 20.04
- ▶ Image docker
- ▶ Reproduction scénario de collecte (Malmö pur)
- ▶ Création de l'interface Malmö/ROS/Overworld
- ▶ Mock-up du «Dialog Manager»

# Tâches réalisées

Point de départ : Stage Ismail (Enregistrement d'états en .json)

- ▶ MàJ Malmö pour fonctionner sur Ubuntu 20.04
- ▶ Image docker
- ▶ Reproduction scénario de collecte (Malmö pur)
- ▶ Création de l'interface Malmö/ROS/Overworld
- ▶ Mock-up du «Dialog Manager»
- ▶ Mock-up du Planner



# Tâches réalisées

Point de départ : Stage Ismail (Enregistrement d'états en .json)

- ▶ MàJ Malmö pour fonctionner sur Ubuntu 20.04
  - ▶ Image docker
  - ▶ Reproduction scénario de collecte (Malmö pur)
  - ▶ Création de l'interface Malmö/ROS/Overworld
  - ▶ Mock-up du «Dialog Manager»
  - ▶ Mock-up du Planner
- } regex FTW!

# Sommaire

## 3. Petite démo

# Sommaire

## 4. Conclusion

# Conclusion

- ▶ Preuve de concept

# Conclusion

- ▶ Preuve de concept
  - ▶ Interaction avec Minecraft utilisant ROS

# Conclusion

- ▶ Preuve de concept
  - ▶ Interaction avec Minecraft utilisant ROS
  - ▶ Intégration partiel des ontologies

# Conclusion

- ▶ Preuve de concept
  - ▶ Interaction avec Minecraft utilisant ROS
  - ▶ Intégration partiel des ontologies
  - ▶ Tâches extensibles par autres tâches

# Conclusion

- ▶ Preuve de concept
  - ▶ Interaction avec Minecraft utilisant ROS
  - ▶ Intégration partiel des ontologies
  - ▶ Tâches extensibles par autres tâches
  - ▶ Plateforme pour expérimentation avec dialogue et ontologies



# Conclusion

- ▶ Preuve de concept
  - ▶ Interaction avec Minecraft utilisant ROS
  - ▶ Intégration partiel des ontologies
  - ▶ Tâches extensibles par autres tâches
  - ▶ Plateforme pour expérimentation avec dialogue et ontologies
- ▶ Dans l'horizon

# Conclusion

- ▶ Preuve de concept
  - ▶ Interaction avec Minecraft utilisant ROS
  - ▶ Intégration partiel des ontologies
  - ▶ Tâches extensibles par autres tâches
  - ▶ Plateforme pour expérimentation avec dialogue et ontologies
- ▶ Dans l'horizon
  - ▶ Parser plus intelligent avec LLM

# Conclusion

- ▶ Preuve de concept
  - ▶ Interaction avec Minecraft utilisant ROS
  - ▶ Intégration partiel des ontologies
  - ▶ Tâches extensibles par autres tâches
  - ▶ Plateforme pour expérimentation avec dialogue et ontologies
- ▶ Dans l'horizon
  - ▶ Parser plus intelligent avec LLM
  - ▶ ajouter réactivité aux changements d'état

# Conclusion

- ▶ Preuve de concept
  - ▶ Interaction avec Minecraft utilisant ROS
  - ▶ Intégration partiel des ontologies
  - ▶ Tâches extensibles par autres tâches
  - ▶ Plateforme pour expérimentation avec dialogue et ontologies
- ▶ Dans l'horizon
  - ▶ Parser plus intelligent avec LLM
  - ▶ ajouter réactivité aux changements d'état
  - ▶ Planification des mouvements (point d'approche etc)

Contact

[rafael.accacio.nogueira@gmail.com](mailto:rafael.accacio.nogueira@gmail.com)

