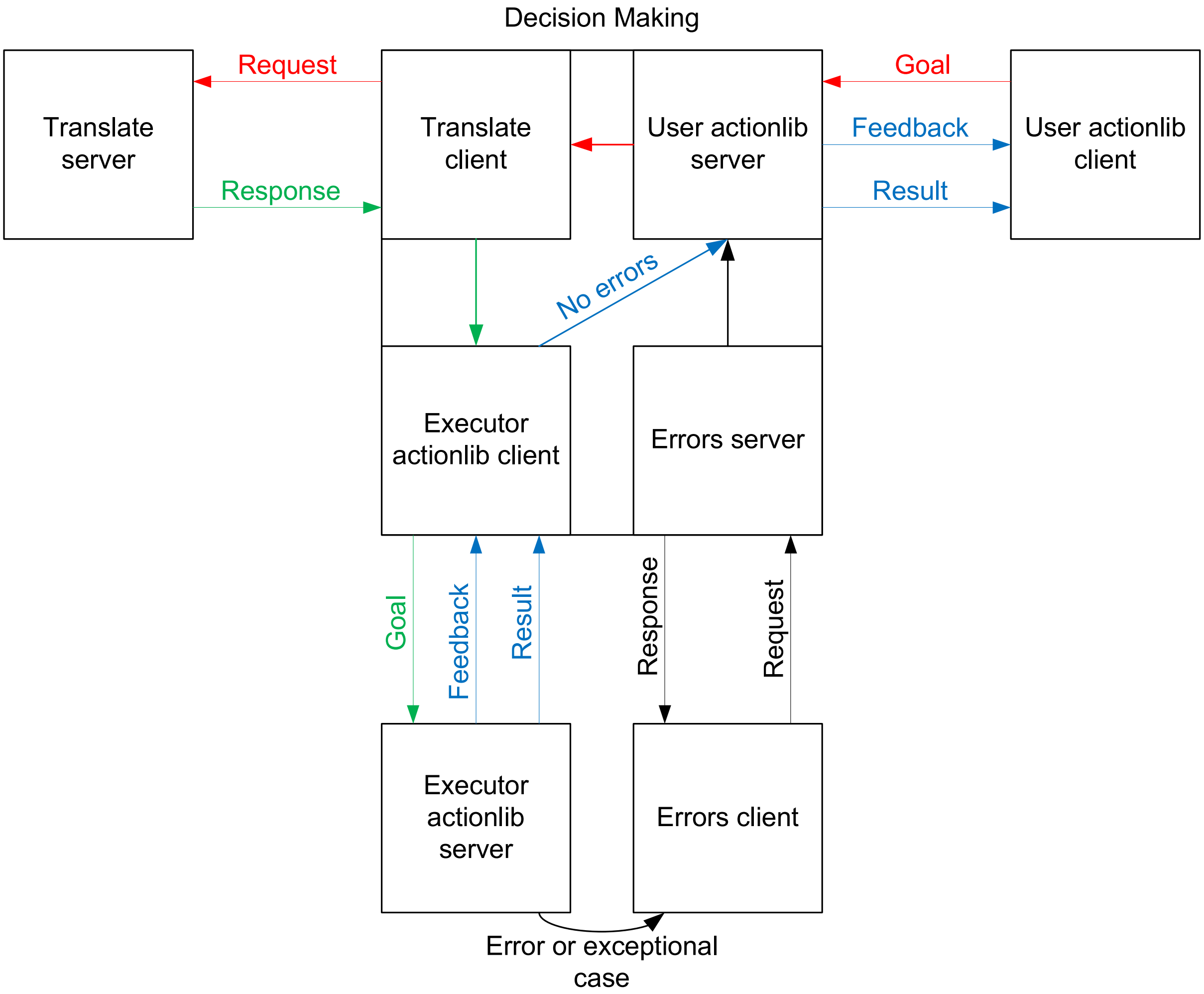
# STACK OF THE PACKAGE : srs_control_task



# SERVICES

## ACTIONLIB

| *Name* | Execution.action | User.action |
|---|---|---|
| *Details* | Interface between the Decision Making and task executor. Decision making send an action with the parameter, then, the server execute the action. | Interface between the Decision Making and user service. The human operator select a command, send it to the decision making, then, wait for feedback and result of the process |
| *Messages* | #define the goal<br>string action<br>string parameter #for example a place or a position<br>---<br>#define the result<br>uint32 return_value<br>---<br>#define a feedback message<br>string current_state<br>bool solution_required<br>uint32 exceptional_case_id | #define the goal<br>string actionName<br>string parameter<br>string solution<br>---<br>#define the result<br>int32 returnValue<br>---<br>#define a feedback message<br>string currentState<br>bool solutionRequired<br>uint32 exeptionalCase_id |

## SERVICE

| *Name* | translation.srv | errors.srv |
|---|---|---|
| *Details* | Interface between the Decision Making and the translator. Later there will be an access to the database to translate the command in a low level control message | Interface between the task executor and the Decision making for an exceptional case. The task executor send different messages to the Decision making for report an error, then, the Decision making send a new target. |
| *Messages* | #define the request<br>string highLevelMess<br>---<br>#define the response<br>string response | #define the request<br>string current_state<br>uint32 exceptional_case_id<br>---<br>#define the response<br>string solution |

## CODE

| File name | Details |
|---|---|
| **Decision_making.py** | Let to make the links between the different services. So in first it receives the command by human operator, then, call the translation service, then, the task execution service. Finally, it return result and feedback about all the sequence to the user. |
| **Translator** | Receives a high level message which is a word and translate it before return it to the decision making. Later there will be a database access |
| **task_executor.py** | Receives the goal furnished by *Execution.action* and trigger the state machine. |
| **high_level_script_server.py** | Supply all the Smach states used to execute a task with the robot. Also contains the states for the exceptional case. |

## BUILD THE PACKAGE

```
roscd srs_control_task

make
```

## RUNNING THE SCRIPT

1. **Launch the simulation**

```
roslaunch srs_get_milk get_milk.launch
```

2. **Start the tasks executor server**

```
rosrun srs_control_task task_executor.py
```

3. **Start the translator server**

```
rosrun srs_control_task translate_server.py
```

4. **Start the user decision making**

```
rosrun srs_control_task decisionMaking.py
```

5. **Start the client user**

```
rosrun srs_control_task user_client.py
```

(after you have to follow the instructions)

6. **Optionnal steps , visualization of the state machine**

```
rosrun smach_viewer smach_viewer.py
```