An Agent-Based Architecture for Context-Aware Communication

Romelia Plesa

School of Information Technology and Engineering,
University of Ottawa, Canada

Luigi Logrippo

Département d'informatique et ingénierie,
Université du Québec en Outaouais

Outline

- Motivation and Proposed architecture
- BDI basics
- Applying BDI
- AgentSpeak(L) basics
- Example
- Conclusion

Motivation

Presence, Context and the personalization of services

- Most of today's telephony communication services could be characterized as *context free*.
  - They provide no real-time context regarding the purpose or the circumstances of a phone call that one is receiving.
- Context information is needed in order to manage the use of the phone or other communication services.
- Context-aware support provides the application relevant knowledge about the environment in which it functions.

Vision

The advocates of presence technology and contextual services promise a world where people will be connected

- ***When they want***
- ***How they want***
- ***With whom they want***

Communication will be tailored on *specific* **desires** and **preferences**.

Goal

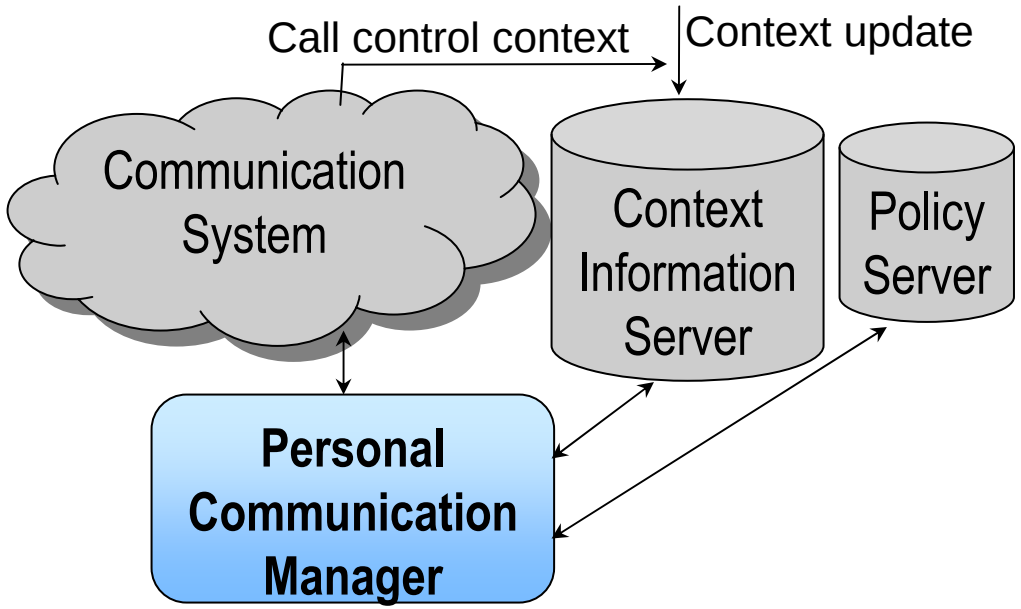
- to propose an architecture that supports presence and contextual services in telecom and allows context-aware call handling based on:

  - information about the environment (context)
  - individual policies.

New Services

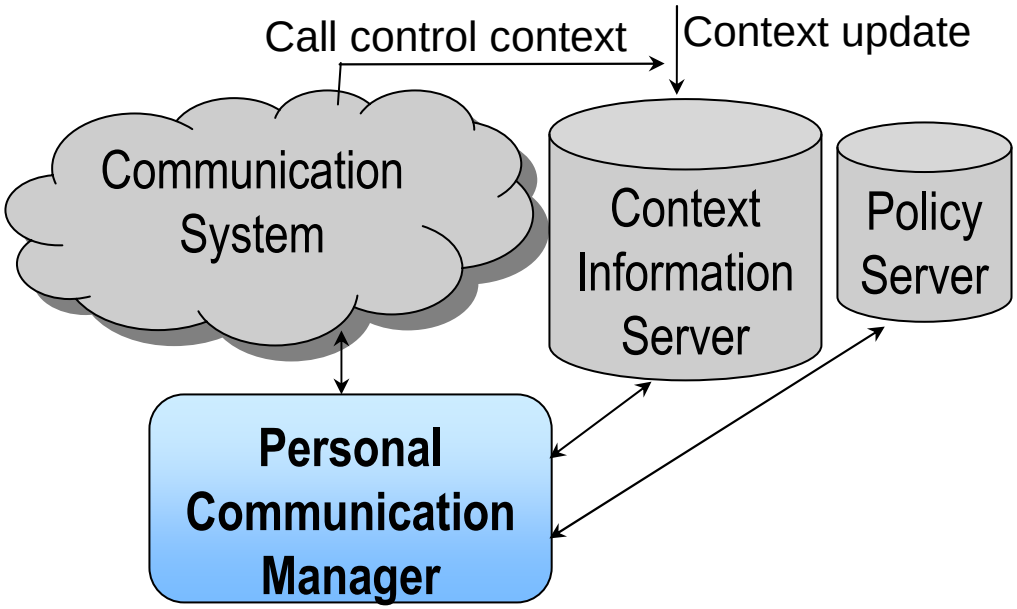
- **Context-based services**
  - ***All calls from my students will have announcement X played out.***
- **Availability services**
  - ***Secretaries are not available to answer enquires during lunchtime***
- **Notification services**
  - ***Remind me of the 3 pm meeting if I am not already in the meeting room.***
- **Personal addressing services**
  - ***If the call is from a person involved in project X, redirect it to the team leader.***

The Architecture
~ Functional Requirements ~



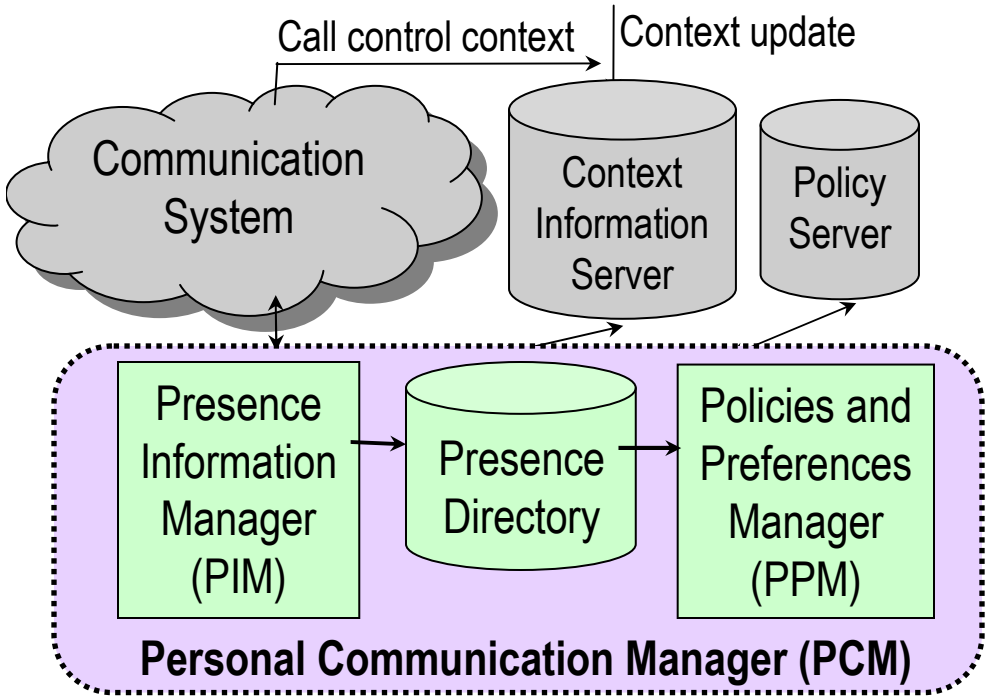
- collection of context information using sensors
- dissemination of context information
- publishing of presence information from users and their devices
- description of user policies and preferences
- user preferences-based handling of communication

The Architecture
~ Characteristics ~



- independent of the communication protocol (SIP, H.323).
- **Context Information Server** updates, stores and distributes the context information.
- **Policy Server** manages the user's personal and subscription / notification policies.

Personal Communication Manager



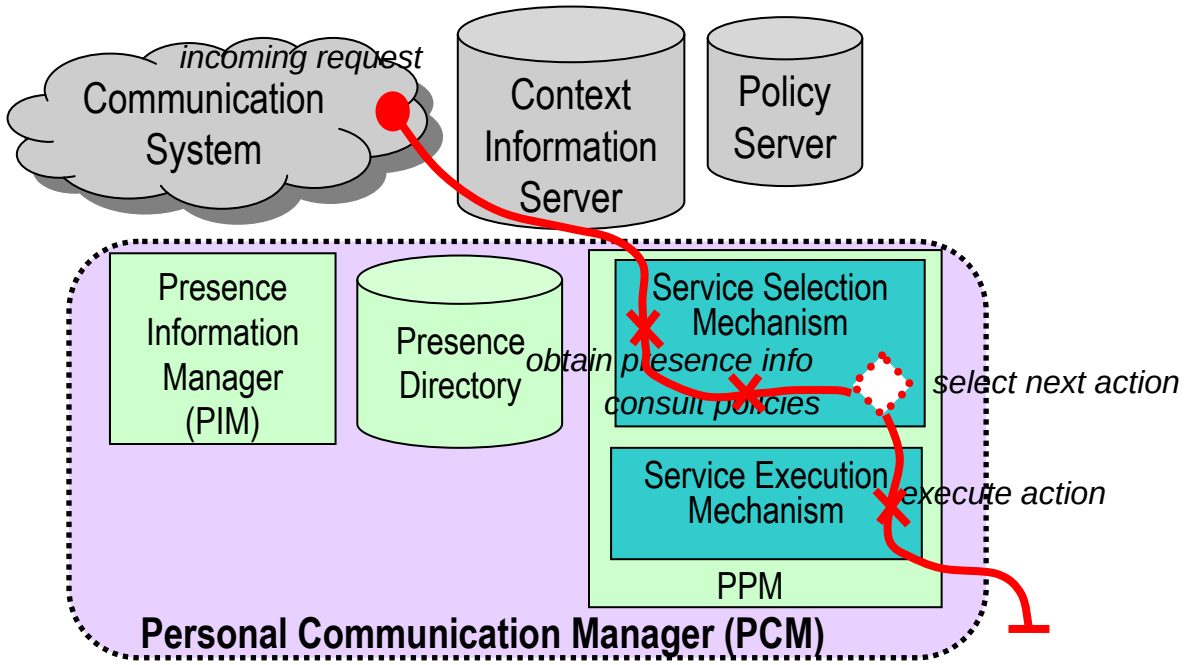
- a *software agent* that represents each user.
- receives request messages (e.g. SIP INVITE) and decides how they should be handled.

– **Presence Information Manager** - a rule-based process that builds the "consolidated presence information".

– **Presence Directory** - a repository in which known and deduced presence information is deposited.

– **Policies and Preferences Manager -** contains the preferences logic to respond to requests to contact an entity.

Context Aware Call Handling



- Includes context update, service selection based on context information and user personal policies as well as service execution.
  - The service selection and execution mechanisms will be incorporated into the Personal Communication Manager (in the **Policies and Preferences Manager (PPM)** component).

The BDI Model

- Belief, Desire, Intention (BDI) is an architecture for modeling Intelligent Software Agents.
- BDI agents can solve problems in dynamic and real-time environments.
- The BDI architecture is used in a variety of applications:
  - robots that play soccer
  - air traffic controllers in airports.

BDI Agents

- Systems that are situated in a changing environment
- Receive continuous perceptual input
- Take actions to affect their environment

From the various options and alternatives available to it at a certain moment in time, the agent needs to select the appropriate actions or procedures to execute.

The ***selection function*** should enable the system to achieve its objectives

BDI Agents

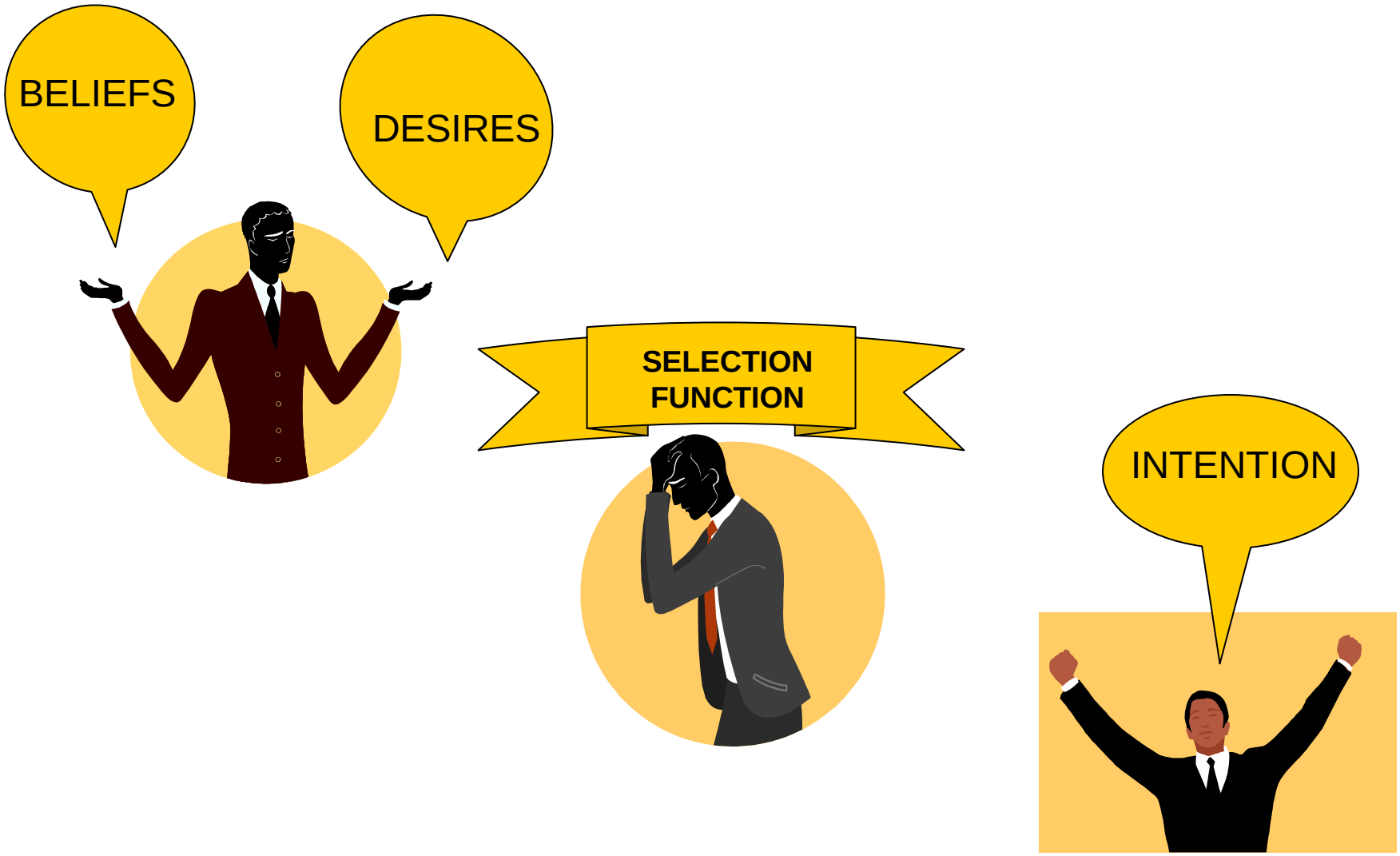
Input:

- **Beliefs:**
  - the characteristics of the environment.
  - updated appropriately after each sensing action.
  - the ***informative*** component.
- **Desires**
  - information about the objectives to be accomplished, the priorities and payoffs associated with the various objectives.
  - the ***motivational*** component.

Output:

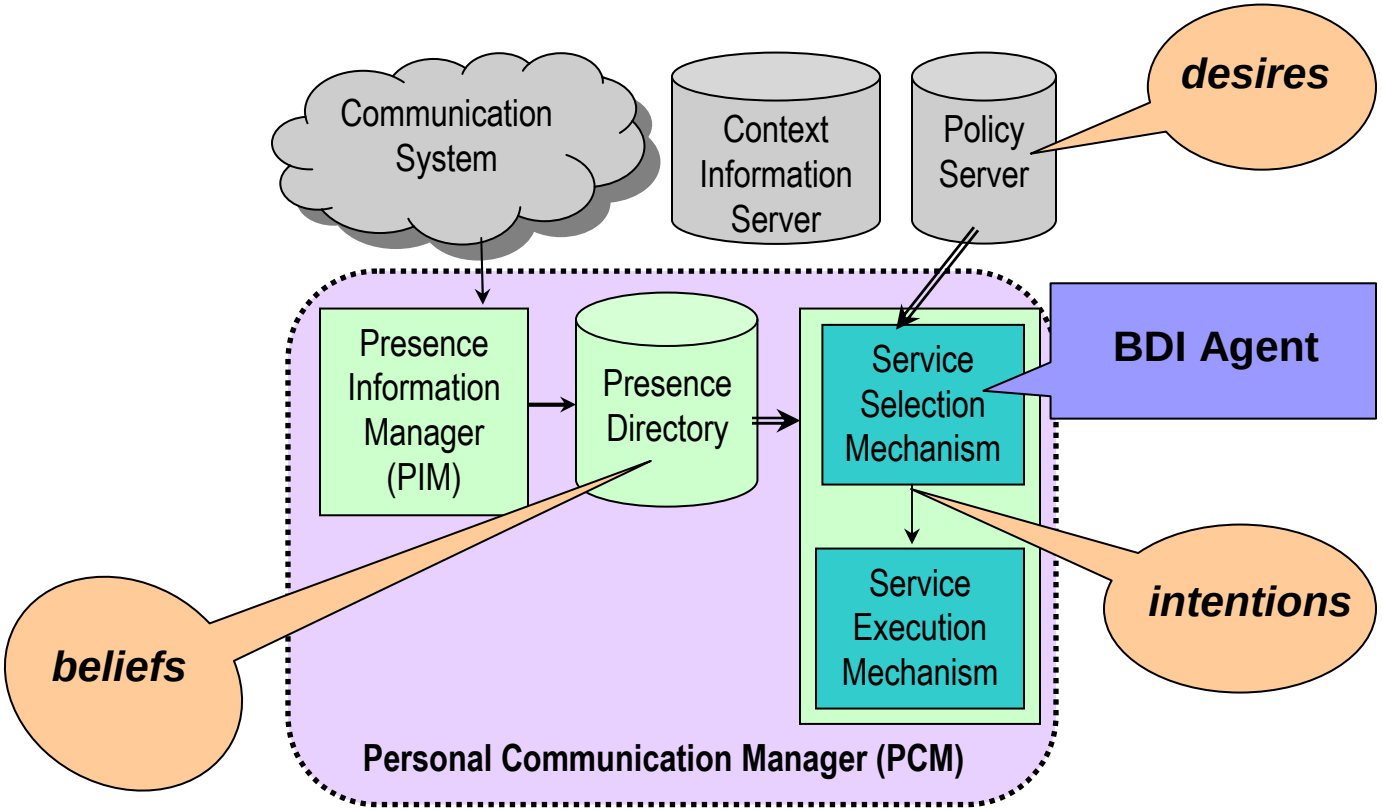
- **Intentions**
  - the currently chosen course of action (the output of the most recent call to the selection function)
  - the ***deliberative*** component.

BDI Agents



BDI Mapping

- implementing PPM as a BDI agent



Representing Context and Policies

- implement PPM as a BDI agent that conforms to the AgentSpeak(L) formalism.
  - abstract framework for programming BDI agents.
  - natural extension of logic programming for the BDI agent architecture.
  - based on a restricted first-order language with events and actions.
- the behavior of the agent (i.e., its interaction with the environment) is dictated by the programs written in AgentSpeak(L).

AgentSpeak(L) - Basic Notions

- The specification of an agent in AgentSpeak(L) consists of:
  - a set of base ***beliefs*** - facts in the logic programming sense
  - a set of ***plans***.
    - context-sensitive, event-invoked.
    - allow hierarchical decomposition of goals and the execution of actions.

```
p ::= te : ct <- h
```

- ***te*** *- triggering event (denoting the purpose for that plan)*
- ***ct*** *- conjunction of belief literals representing a context.*
- ***h*** *- a sequence of basic actions or (sub)goals that the agent has to achieve (or test) when the plan, if applicable, is chosen for execution.*

AgentSpeak(L) - Basic Notions

- ***goal***
  - s a state of the system, which the agent wants to achieve.
  - ***achievement goals*** - the agent wants to achieve a state of the world where the associated predicate is true.
  - ***test goals*** - returns a unification with one of the agent's beliefs; it fails if no unification is found.
- ***triggering event***
  - defines which events may initiate the execution of a plan.
  - internal, when a subgoal needs to be achieved
  - external, when generated from belief updates

AgentSpeak(L) - Basic Notions

- ***Intentions***
  - plans the agent has chosen for execution.
  - executed one step at a time.
  - a step can
    - query or change the beliefs
    - perform actions on the external world
    - suspend the execution until a certain condition is met
    - submit new goals.
  - the operations performed by a step may generate new events, which, in turn, may start new intentions.
  - an intention succeeds when all its steps have been completed. It fails when certain conditions are not met or actions being performed report errors.

AgentSpeak(L) Example

Alice

(1) During lunch time, forward all calls to Carla.

(2) When I am busy, incoming calls from colleagues should be forwarded to Denise.

Beliefs

```
user(alice).
user(bob).
user(carla).
user(denise).
~status(alice, idle).
status(bob, idle).
colleague(bob).
lunch_time("11:30").
```

AgentSpeak(L) Example

```
user(alice).
user(bob).
user(carla).
user(denise).
~status(alice, idle).
status(bob, idle).
colleague(bob).
lunch_time("11:30").
```

"During lunch time, forward all calls to Carla".

```
+invite(X, alice) : lunch_time(t)   ←
  !call_forward(alice, X, carla). (p1)
```

"When I am busy, incoming calls from colleagues should be forwarded to Denise".

```
+invite(X, alice) :
    colleague(X)  ←
  call_forward_busy(alice,X,denise).
                                  (p2)
```

```
+invite(X, Y): true   ←  connect(X,Y).
                                  (p3)
```

AgentSpeak(L) Example

```
user(alice).
user(bob).
user(carla).
user(denise).
~status(alice, idle).
status(bob, idle).
colleague(bob).
lunch_time("11:30").
+invite(X, alice) : lunch_time(t)   ←   !call_forward(alice, X, carla).   (p1)
+invite(X, alice) :     colleague(X)  ← call_forward_busy(alice,X,denise).(p2)
+invite(X, Y): true   ←  connect(X,Y).                                     (p3)
```

```
+!call_forward(X, From, To) : invite(From, X)
  ← +invite(From, To), - invite(From,X)          (p4)
```

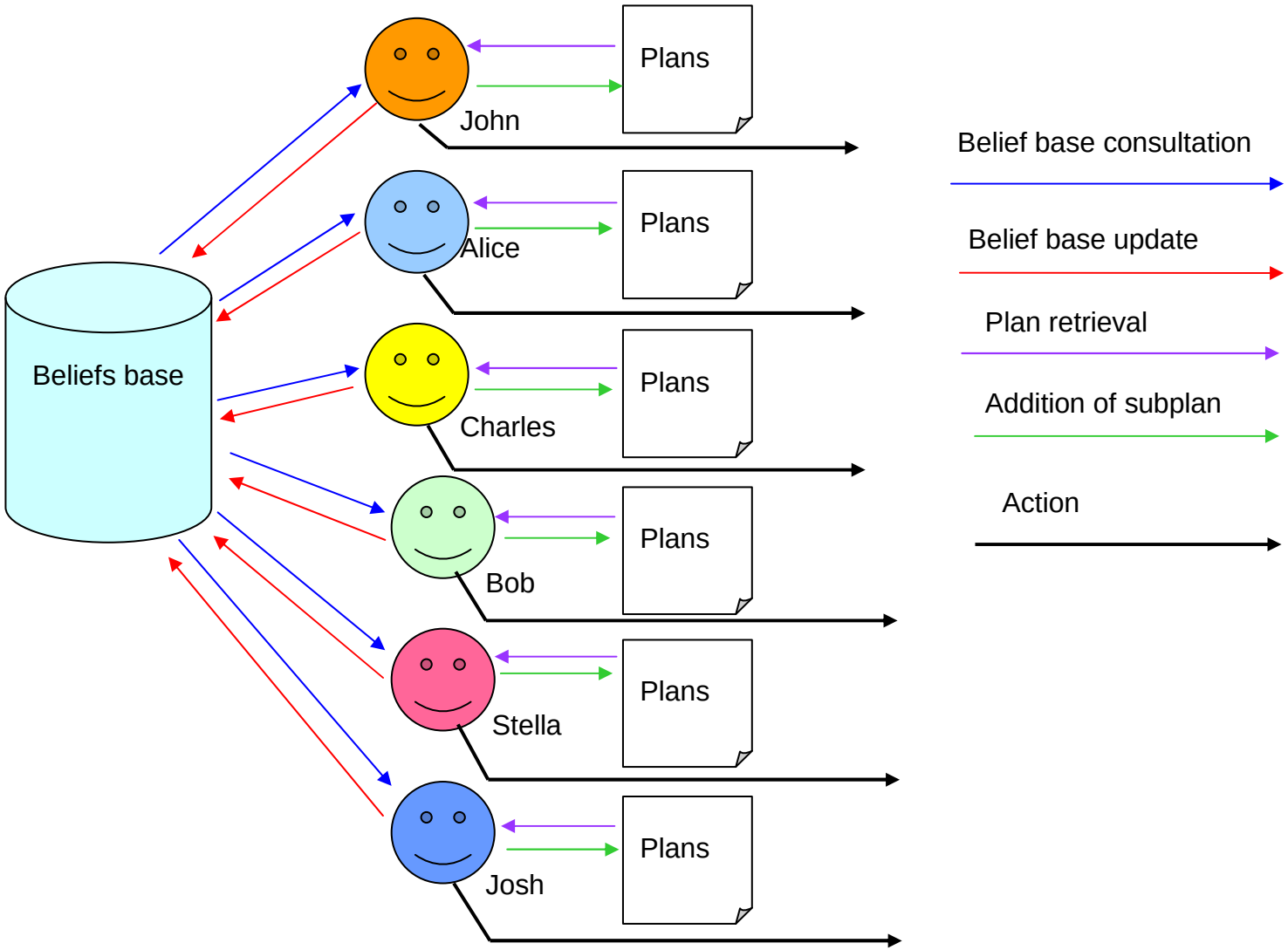
```
+!call_forvard_busy(Y, From, To) : invite(From, Y)&
  not(status(Y, idle)))
      ← +invite(From, To), - invite(From,Y).     (p5)
```

AgentSpeak(L) Example

```
user(alice).
user(bob).
user(carla).
user(denise).
~status(alice, idle).
status(bob, idle).
colleague(bob).
lunch_time("11:30").

+invite(X, alice) : lunch_time(t)
                ← !call_forward(alice, X, carla).            (p1)
+invite(X, alice) :     colleague(X)
                ← call_forward_busy(alice,X,denise).         (p2)
+invite(X, Y): true   ←  connect(X,Y).                      (p3)
+!call_forward(X, From, To) : invite(From, X)
   ← +invite(From, To), - invite(From,X)                          (p4)
+!call_forvard_busy(Y, From, To) : invite(From, Y)& not(status(Y,
   idle)))
        ← +invite(From, To), - invite(From,Y).                (p5)
```

Simulation


Plans
John
Plans
Alice
Beliefs base
Plans
Charles
Plans
Bob
Plans
Stella
Plans
Josh
Belief base consultation
Belief base update
Plan retrieval
Addition of subplan
Action

Essential features (1)

- **Context-sensitivity**
  - The beliefs base is updated with all the changes in the environment using the AgentSpeak mechanism of event perception.
- **Plan selection**
  - If multiple applicable plans are available, the agent is able to select one that is appropriate.
  - May depend on the time needed, the overall cost, the risk factor, the user preferences, etc.
  - Decision procedures must therefore be supplied for supporting plan selection.

Essential features (2)

- **Plan failure recovery**
  - If a plan fails at some stage, the agent is able to retract properly and select another alternative plan.
- **Conflict resolution and goal selection**
  - The user might have a number of goals that cannot be achieved simultaneously.
  - In such cases, the agent must be able to make a decision about which goals to try to achieve.
  - In making such decisions, it needs to take into account the importance of the goals as well as the costs of executing the plans.

Conclusions

- The BDI agent paradigm, although originally developed for other purposes, is particularly suited to the user communication domain.
- The actions that the agent decides to take arise from the instantiation of partially specified plans, selected to fulfill the user's goals, given the beliefs that it has at that point in time.
- The details of the plan are filled in as the plan progresses, which allows for a wide range of possible courses of action.