

Sebastian Sardina and Lin Padgham

RMIT University

{ssardina,linpa}@cs.rmit.edu.au

MOTIVATION & OBJECTIVES

BDI-style **agent-oriented programming** is a novel approach to programming complex systems in dynamic environments.

These systems are very open to changes in the environment, flexible, and robust. Thus, they are suited for soft real-time reasoning and control.

They are build around the concept of **goals**.

However, the level of support for goals has not been commensurate with their importance:

- mostly procedural view;
- weak representation and reasoning;
- not well linked with agent-theories.

What do we do in this paper?

We substantially improve the handling of goals in the BDI-style language CANPLAN, which includes built-in **failure handling**, **declarative goals**, and **HTN-style planning**.

CANPLAN2 = "flexible single-minded" CANPLAN

BDI-PROGRAMMING

A very popular **agent-oriented** programming methodology with roots in philosophy and folk-psychology.

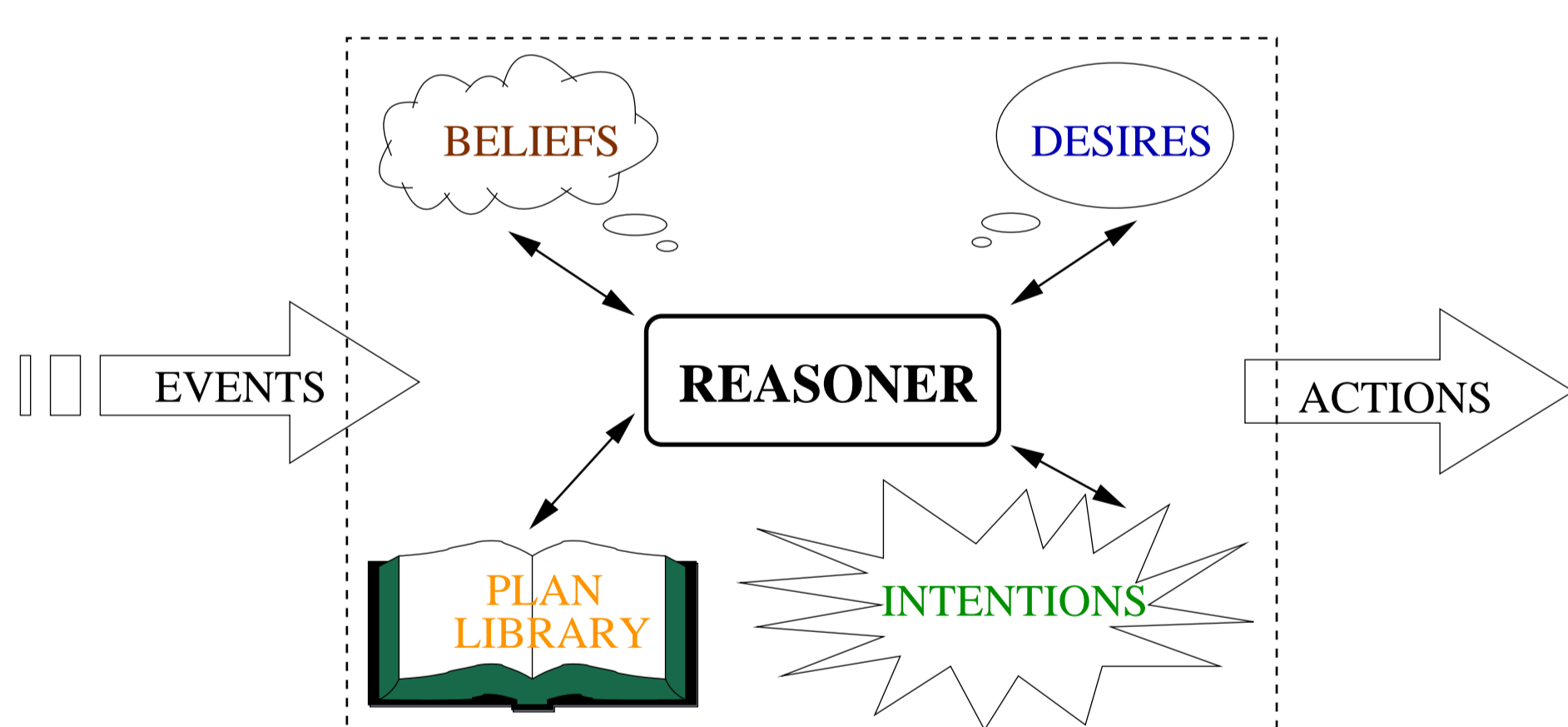
Key features of BDI agent-oriented systems:

Beliefs: information about the world (set of facts).

Events: goals/desires to resolve; internal or external.

Plan library: recipes for handling goals-events: $e : \psi \leftarrow P$.

Intentions: partially instantiated programs; commitment.



Behavior arises due to the agent **committing** to some of its **desires**, and selecting **actions** that achieve its **intentions** given its **beliefs**.

Examples of BDI-style languages/platforms:

AGENT SPEAK/JASON
3APL/GOAL
JACK

PRS & DMARS
JAM
CAN(PLAN)

THE BDI EXECUTION CYCLE [RAO&GEORGEFF 92]

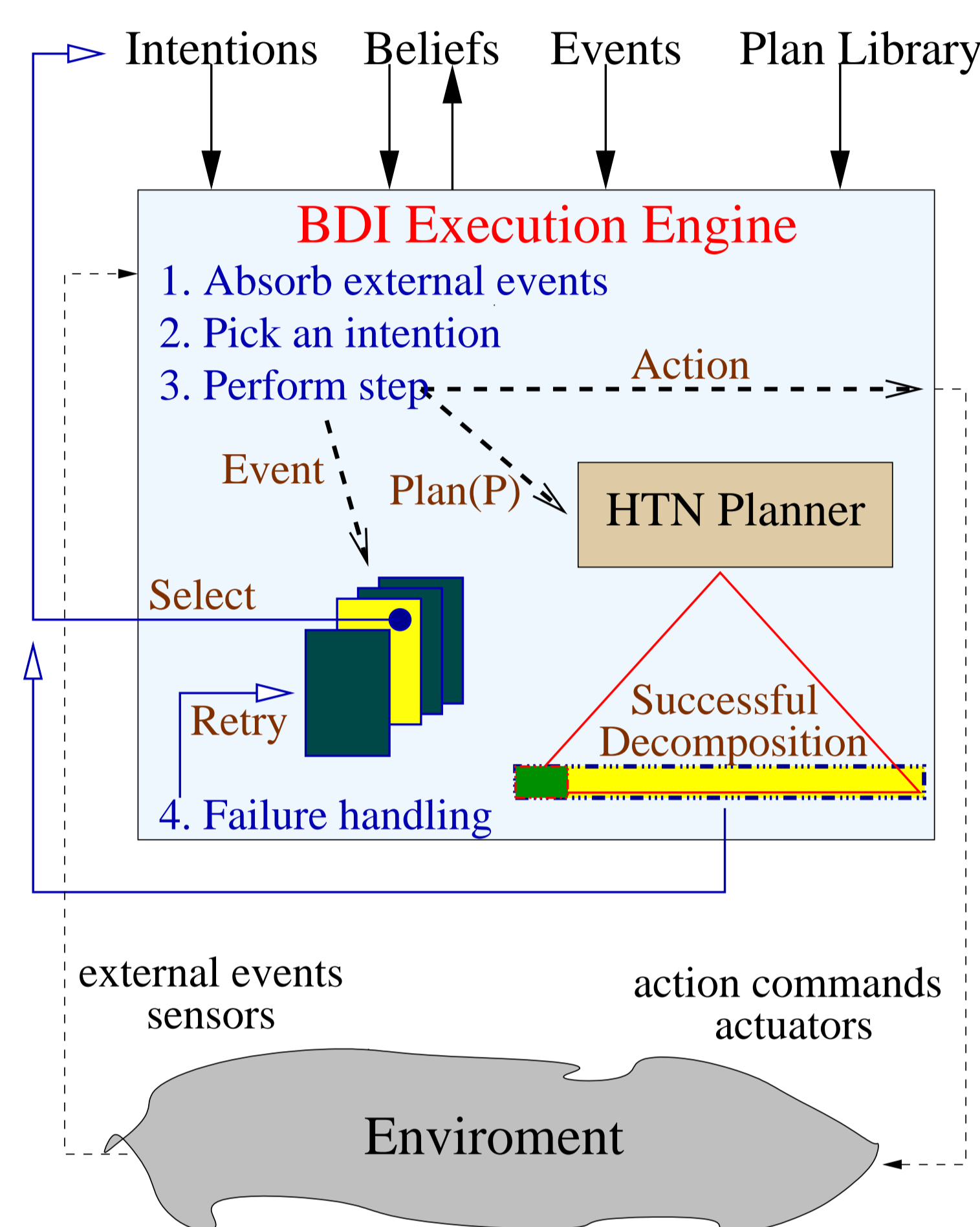
1. Pick a **pending event** e .
2. Select **relevant** plans from library (match event).
3. Select **applicable** plans from relevant set (match context).
 - (a) If e is *external*, create new intention with selected plan.
 - (b) If e is *internal*, update the corresponding intention.
4. Select and partially **execute** an intention:
 - (a) Execution of the intention may post new (internal) events.
 - (b) If execution is impossible, then perform **failure recovery**.
5. **Observe** the environment for new *external* events.
6. **Update the set of goals and intentions**.
7. Repeat cycle.

CANPLAN= BDI + PLANNING

CAN \approx AGENT SPEAK + Goal(ϕ_s, P, ϕ_f) + Fail. Hand.

CANPLAN = CAN + HTN-style local planning

Architecture Overview



The Language

A CANPLAN agent is defined as $C = \langle \mathcal{N}, \Pi, \mathcal{B}, \mathcal{A}, \Gamma \rangle$, where:

1. \mathcal{N} is the **agent name**.

2. Π is a **plan library** containing plan rules $e : \psi \leftarrow P$
 e is the *triggering event* and ψ is the *context condition*;
 P is the *plan-body* from the following language:

$act/?\phi,$	primitive action/test
$+b, -b$	add/delete belief atom
$!e$	event goal
$P_1; P_2$	sequence
$P_1 P_2$	interleaved concurrency
$Goal(\phi_s, !e, \phi_f)$	declarative goal
$Plan(P_1)$	HTN-style planning

3. \mathcal{B} is the **belief base**: current agent's knowledge.

4. \mathcal{A} is the sequence of **executed actions**.

5. Γ is the **intention base**: partially instantiated plan-bodies.

Operational Semantics

1. Intention-level transitions: $\langle \mathcal{B}, \mathcal{A}, P \rangle \xrightarrow{bdi/plan} \langle \mathcal{B}', \mathcal{A}', P' \rangle$

2. Agent-level transition: $\langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \Gamma \rangle \Rightarrow \langle \mathcal{N}, \Lambda, \Pi, \mathcal{B}', \mathcal{A}', \Gamma' \rangle$

$!e \rightarrow \Delta \rightarrow P_i \triangleright \Delta' \xrightarrow{*} fail \triangleright \Delta' \rightarrow \Delta' \rightarrow \dots$

$Goal(\phi_s, !e, \phi_f) \rightarrow Goal(\phi_s, \Delta, \phi_f) \rightarrow Goal(\phi_s, P_i \triangleright \Delta', \phi_f)$

where $\Delta = (\psi_1 : P_1, \dots, \psi_n : P_n)$, and $\Delta' = \Delta \setminus \{\psi_i : P_i\}$.

LIMITATIONS IN CANPLAN

Declarative goals $Goal(\phi_s, P, \phi_f) \dots$

1. are **only adopted** due to the the execution of plans;
2. are **not forced to terminate** when achieved or impossible;
... depends on corresponding intention being selected.
3. are **too "fanatic"** on their procedural methods.
... G_2 is a *blocked subgoal* of G_1 , but G_1 has an *alternative*.

ENHANCING SUPPORT FOR GOALS

1. Declarative goals can be generated pro-actively.
2. Achieved and "impossible" goals are dropped at every step.
3. Non-working declarative subgoals may be dropped for the sake of achieving (higher-level) motivating goals;
 \Rightarrow differentiate **goals** from their **subgoals**.

4. Only "reactive" intentions can be dropped when blocked;
 \Rightarrow differentiate **event-goals** from declarative **goals**.

Restriction: incremental & modular extension to CANPLAN.
 \Rightarrow do not re-do the language!

SUMMARY OF CHANGES TO CANPLAN

- Introduced a new "**motivational**" library \mathcal{M} to generate goals pro-actively: $\psi \rightsquigarrow Goal(\phi_s, !e, \phi_f) \in \mathcal{M}$
 $RoomDirty \wedge \neg Busy \rightsquigarrow Goal(\neg RoomDirty, !clean, HasWork)$.
Agent configurations are now $\langle \mathcal{N}, \mathcal{M}, \Lambda, \Pi, \mathcal{B}, \mathcal{A}, \Gamma \rangle$.

- Divided the **agent-level semantics** (\xrightarrow{agent}) in **three-phases**:

1. intention execution (\xrightarrow{int});
2. handling of external events & sensor information (\xrightarrow{ev});
3. goal update (\xrightarrow{goal}).

- Introduced 9 new and modified 4 **derivation rules**.
 \Rightarrow implement BDI rational execution cycle.

$$\frac{C \xrightarrow{int} C' \quad C' \xrightarrow{event} C'' \quad C'' \xrightarrow{goal} C'''}{C \xrightarrow{agent} C'''} Agent^1$$

- Defined and used the following **notions**:

- reactive/proactive/planning goals & intentions;
- current & alternatives strategies;
- active goals & active goal traces (from intention base);
- external environment \mathcal{E} .

TECHNICAL RESULTS

Single-Minded Commitment

Agents would not pursue goals that are achieved or deemed unachievable.

Definition 1 An agent configuration C is "**single-minded**" if for every active declarative goal $G(\phi_s, \phi_f)$, $\mathcal{B} \models \phi_s$ and $\mathcal{B} \not\models \phi_f$.

Theorem 1 The single-minded property is **propagated** throughout BDI executions.

Goals and Subgoals

Agents always respect the hierarchical structure of active goals; subgoals are mere instruments for higher level goals.

Theorem 2 Let G_k be an active (sub)goal such that its **current strategy is blocked**. Then, for every **subgoal** $G_{k'}$ of G_k , either:

1. the current strategy for $G_{k'}$ is blocked and $G_{k'}$ does not have an alternative strategy; or
2. there is a planning (sub)goal G_p "between" G_k and $G_{k'}$ with no solution.

Flexible Commitment

All reasons why a declarative goal may be abandoned.

Theorem 3 Assume $C \xrightarrow{agent} C'$ and that $G_k = G(\phi_s, \phi_f)$ is **active in C but not in C'** . Then, one of the following cases must apply:

1. $\mathcal{B}' \models \phi_s$, i.e., the goal has been achieved;
2. $\mathcal{B}' \models \phi_f$, i.e., the goal is believed to be impossible; or
3. there is a **motivating goal** $G_{k'}$ of G_k , such that either:

- $G_{k'} = Goal(\phi'_s, P', \phi'_f)$ and $\mathcal{B}' \models \phi'_s \vee \phi'_f$;
- $G_{k'}$ is a **planning goal** with no solution; or
- $G_{k'}$ is **blocked**, but $G_{k'}$ has an **alternative applicable strategy**.

"Flexible single-minded" =
single-minded + opt. goal reconsideration when blocked
... lays between the simple **single-minded** and the sophisticated **open-minded** commitment strategies.

FUTURE WORK

1. Goals adopted without checking for negative or positive interactions with current active goals.
2. Goals & motivations restricted to achievement goals only.
3. Intentions & goals cannot be "suspended."
4. Planning goals not repaired upon failure.