An Industrial Knowledge Representation for Kit Building Applications

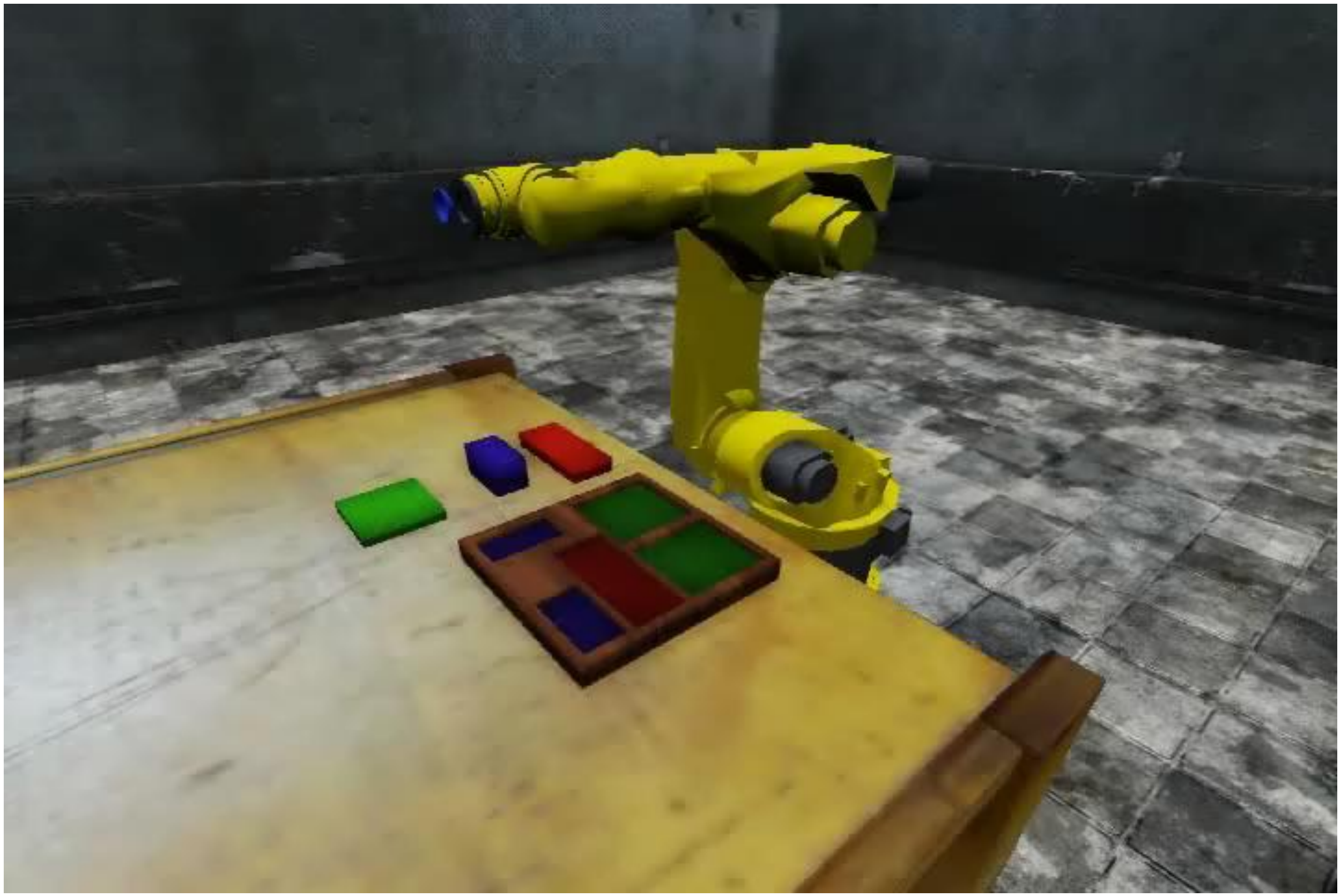
S. Balakirsky, Z. Kootbally, C. Schlenoff, T. Kramer, S. Gupta

Presented by: Stephen Balakirsky

Kit Building

- Process in which individually separate but related items are grouped, packaged, and supplied together as one unit
- Utilized in many manufacturing assembly lines
- Robotic solution requires:
  - Flexibility –Many parts with varying characteristics, part-to-part inconsistences
  - Agility – Changes in part supply locations, lack of fixturing
  - Rapid re-tasking – Ability to quickly build new varieties of kits

Real Planning / Simulated Robot

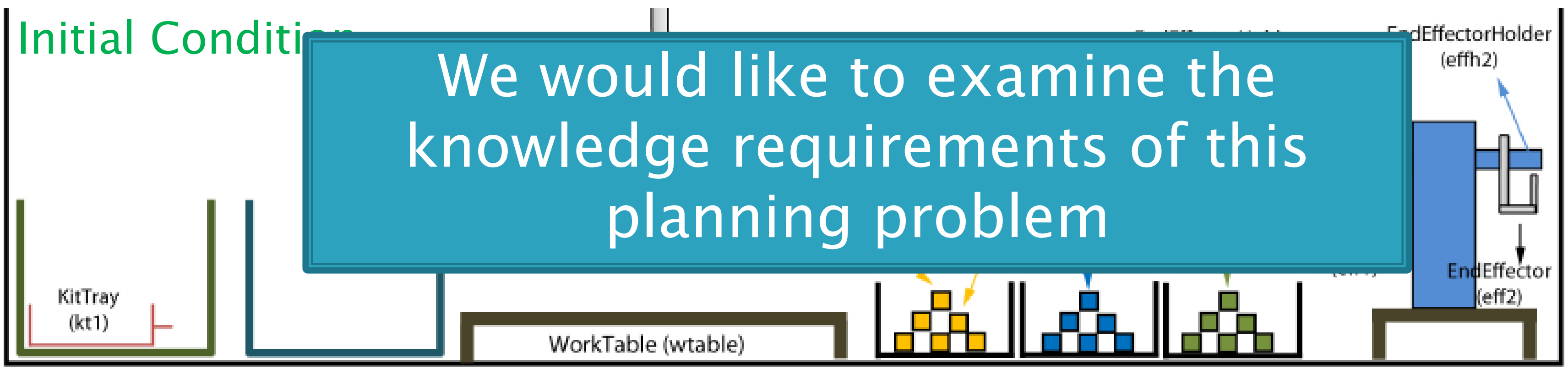


Video is 2x real time

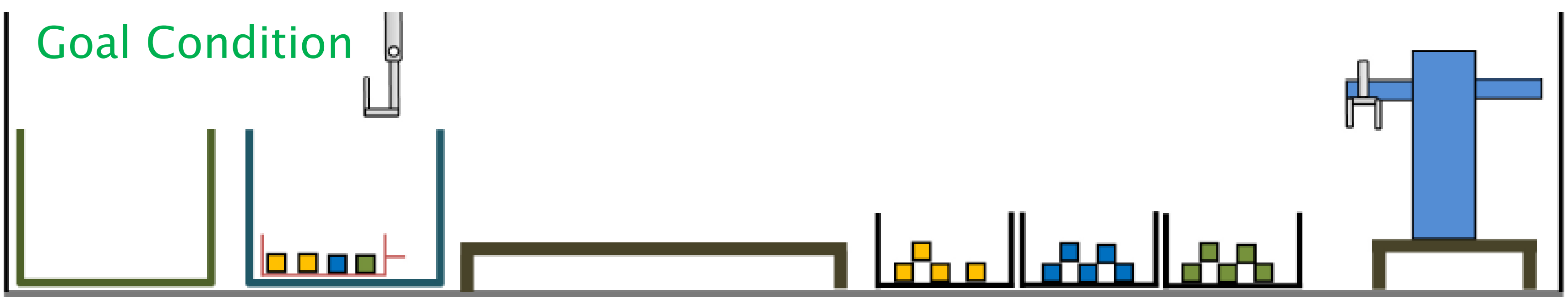
Planning Problem

$$\mathcal{P} = (\Sigma, s_0, g)$$

- Σ – State transition system (possible actions)
- S_0 – Initial conditions (objects and relationships)
- g – Goal conditions



We would like to examine the knowledge requirements of this planning problem



Underlying Knowledge Representation

- Objects
  - What objects and attributes are relevant
  - Taxonomy of objects
  - Relationships between objects
- Actions
  - What actions and attributes are relevant
  - Necessary conditions for an action to occur
  - Likely results of the action
- Dynamic world
  - Object attributes change over time
  - Actions do not always accomplish what we want them to

Objects, Relationships, and Actions

- State-Variable Representation[1] (SVR) used to formally define environment's objects, object relations (state), and actions
- Each state is represented by a tuple of values of *n* state variables $\{x_1, \ldots, x_n\}$
- Action is represented by a partial function that maps this tuple into some other tuple of values of the *n* state variables
- SVR is not a standard interchange language
  - Can encode the SVR into Planning Domain Definition Language (PDDL)

[1]Dana Nau, Malik Ghallab, and Paolo Traverso. 2004. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc.

Planning Domain Definition Language (PDDL)

- Entire kitting domain composed of:
  - 28 predicate expressions that act as preconditions or are acted upon during effects: *rhold-empty, r-with-eff, kit-tray-location, worktable-empty, …*
  - 9 high-level operators (actions): *take-kit-tray*,

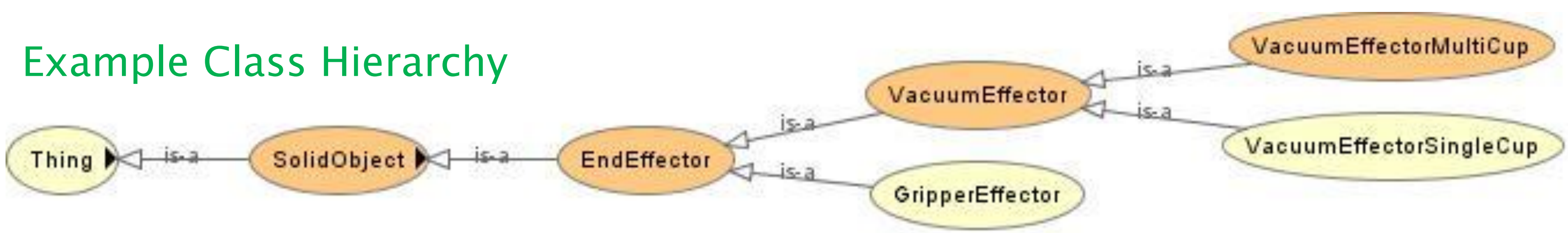
take-kit-tray(r,kt,$lbwekt$,eff,$wtable$): The *Robot* r equipped with the *EndEffector* eff picks up the *KitTray* kt from the *LargeBoxWithEmptyKitTrays* $lbwekt$.

precond	*effects*
rhold-empty(r),	¬rhold-empty(r),
lbwekt-not-empty($lbwekt$),	kit-tray-location(kt,r),
r-with-eff(r,eff),	rhold(r,kt),
kit-tray-location(kt,$lbwekt$),	¬kit-tray-location(kt,$lbwekt$)
eff-location(eff,r),	
worktable-empty($wtable$),	
efftype(eff,kt)	

SVR – PDDL Limitations

- Planning language not designed for knowledge representation
  - Lacks a taxonomy of objects
  - No representation of relationships or constraints between objects
- An ontology provides for all of the above through entities, data properties, and object properties
  - Web Ontology Language (OWL) can be used to represent objects, relationships, and constraints
  - OWL Services (OWL-S) and the Semantic Web Rule Language (SWRL) can be used to represent actions, preconditions, and effects
- Tools exist to move from OWL and OWL-S/SWRL to PDDL representations

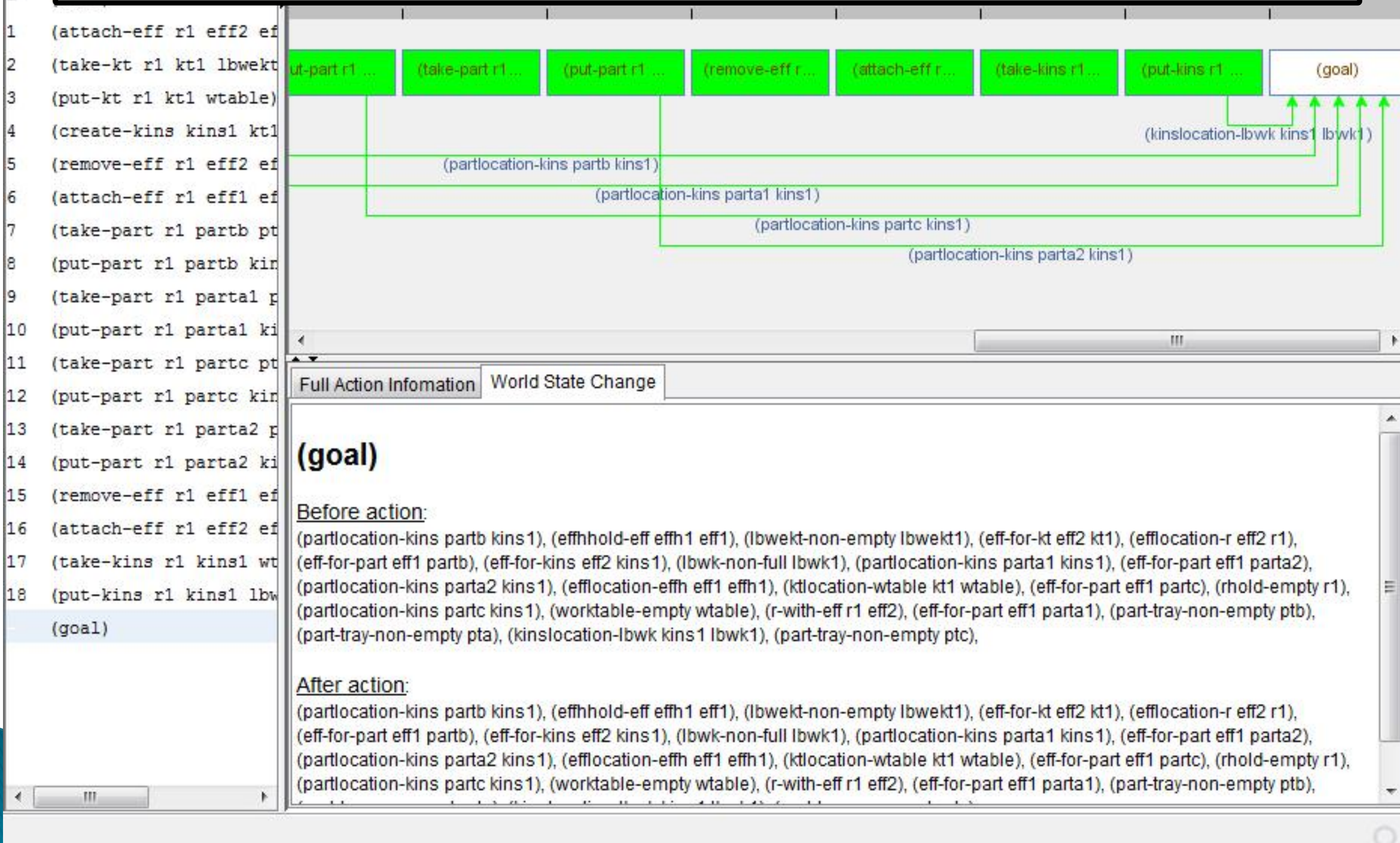
Example Class Hierarchy



Dynamics

- Set of static files with no real-time information
  - Can auto-generate MySQL database from OWL instance file and maintain database with sensor processing
  - Can re-generate OWL instance file to allow for continued reasoning
- Now have a hybrid knowledge structure
  - OWL, OWL-S/SWRL for static snapshot of information
  - MySQL for dynamic knowledge

Planning System in Action



PDDL Plan Instance File

(attach-eff r1 eff2 effh2)

Before action:

(ktlocation-lbwekt kt1 lbwekt1), (effhhold-eff effh1 eff1), ~~(effhhold-eff effh2 eff2)~~, (lbwekt-non-empty lbwekt1), (eff-for-kt eff2 kt1), (eff-for-part eff1 partb), (eff-for-kins eff2 kins1), (lbwk-non-full lbwk1), ~~(efflocation-effh eff2 effh2)~~, (eff-for-part eff1 parta2), (efflocation-effh eff1 effh1), (partlocation-pt partb ptb), (eff-for-part eff1 partc), ~~(r-no-eff r1)~~, (worktable-empty wtable), (partlocation-pt partc ptc), (partlocation-pt parta2 pta), (partlocation-pt parta1 pta), (eff-for-part eff1 parta1), (part-tray-non-empty ptb), (part-tray-non-empty pta), (part-tray-non-empty ptc),

After action:

(ktlocation-lbwekt kt1 lbwekt1), (effhhold-eff effh1 eff1), (lbwekt-non-empty lbwekt1), (eff-for-kt eff2 kt1), (efflocation-r eff2 r1), (eff-for-part eff1 partb), (eff-for-kins eff2 kins1), (lbwk-non-full lbwk1), (eff-for-part eff1 parta2), (efflocation-effh eff1 effh1), (partlocation-pt partb ptb), (eff-for-part eff1 partc), (rhold-empty r1), (worktable-empty wtable), (r-with-eff r1 eff2), (partlocation-pt partc ptc), (partlocation-pt parta2 pta), (partlocation-pt parta1 pta), (eff-for-part eff1 parta1), (part-tray-non-empty ptb), (part-tray-non-empty pta), (part-tray-non-empty ptc),

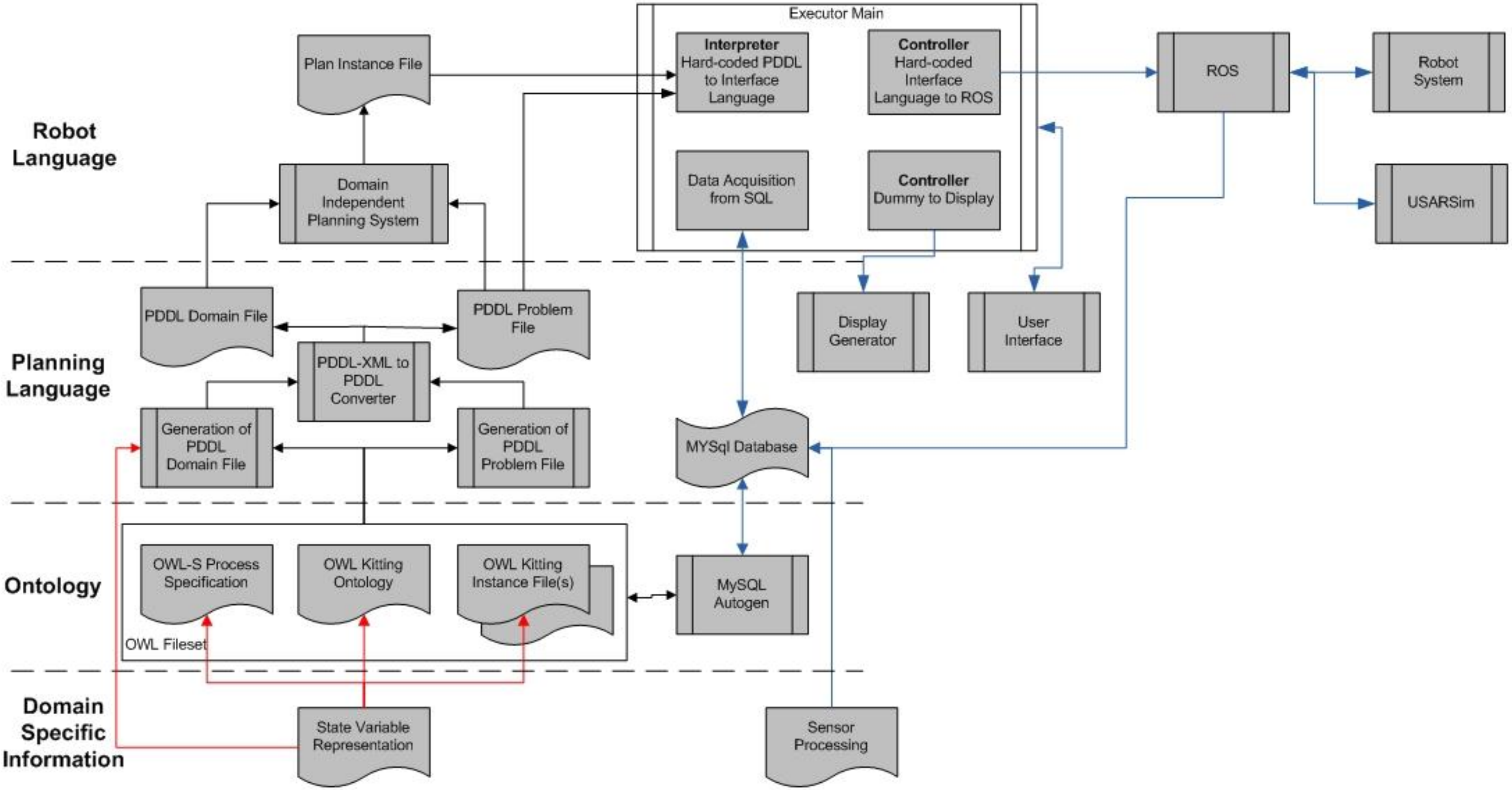
- Intentionally incomplete information
  - Command provides task-level information; most robots will not know how to “attach-eff”
  - Specific information needed to accomplish task is missing
  - This intentional lack of information provides for decoupling of task knowledge and environment knowledge
- Executor module populates the details
  - Combines task knowledge from PDDL with environment knowledge from MySQL
  - Utilizes “Canonical Robot Command Language” as an Interlingua

Example: attach-eff r1 eff2 effh2

Subcommand	PDDL	MySQL	Data Update
Move	R1, effh2	Actual locations, offsets	---
Actuate	R1		R1 holds eff2 effh2 empty
Move	R1	Offsets	---

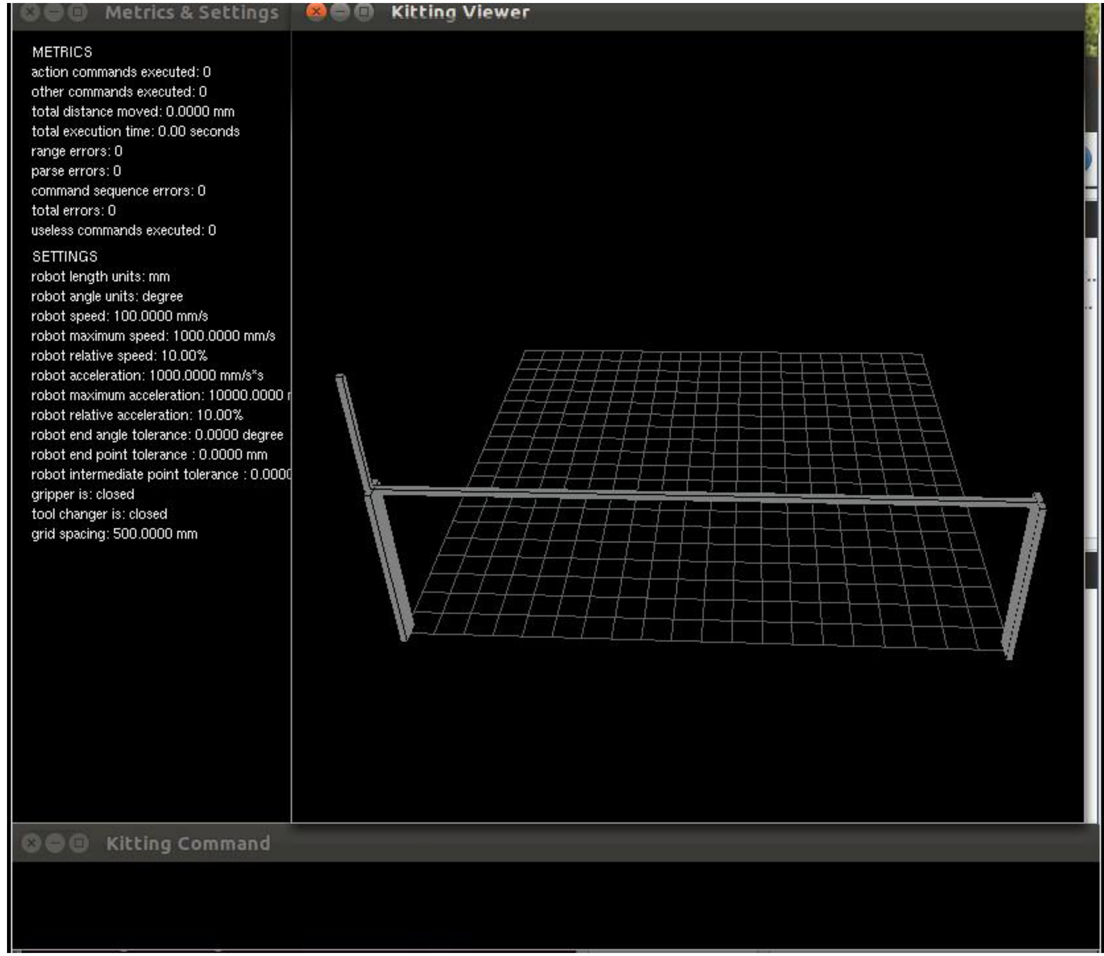
- Must have mapping between PDDL actions and required robot sub-actions
- Must update knowledge base (MySQL)
- MySQL knowledge base contains all data from ontology needed by planner

Complete System Architecture



Future Work, Metrics

- Time to build
- Distance robot traveled
- Number of actions
- Useless commands
- Errors
- …



Future Work

- Current work assumes perfect actions; need techniques to gracefully cope with errors
- Migrate techniques onto real hardware with real image processing
- Extend work to apply to general assembly operations