O_2S Planning Technology: A HOWTO Guide

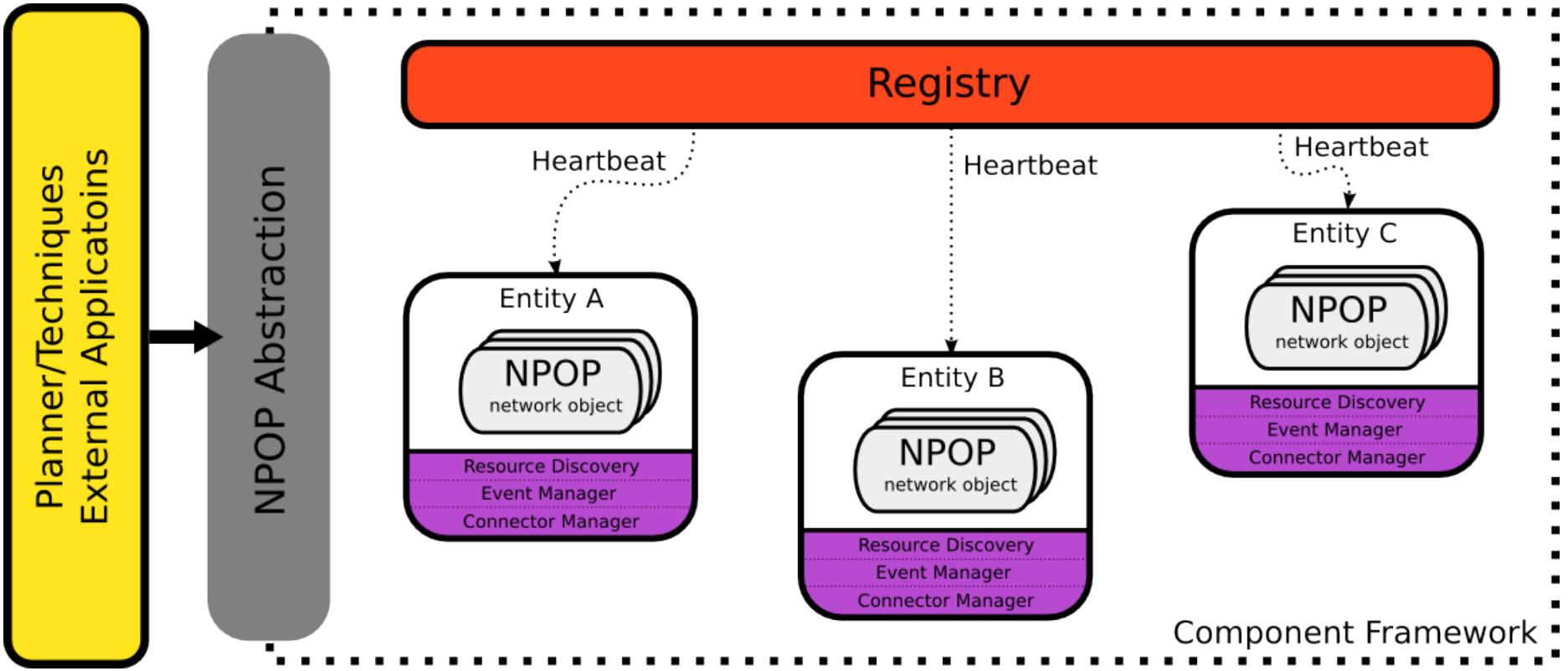
Day 2, Session 2
July 2006
Quanta JustPlay Seminar

Justin Mazzola Paluska
jmp@mit.edu

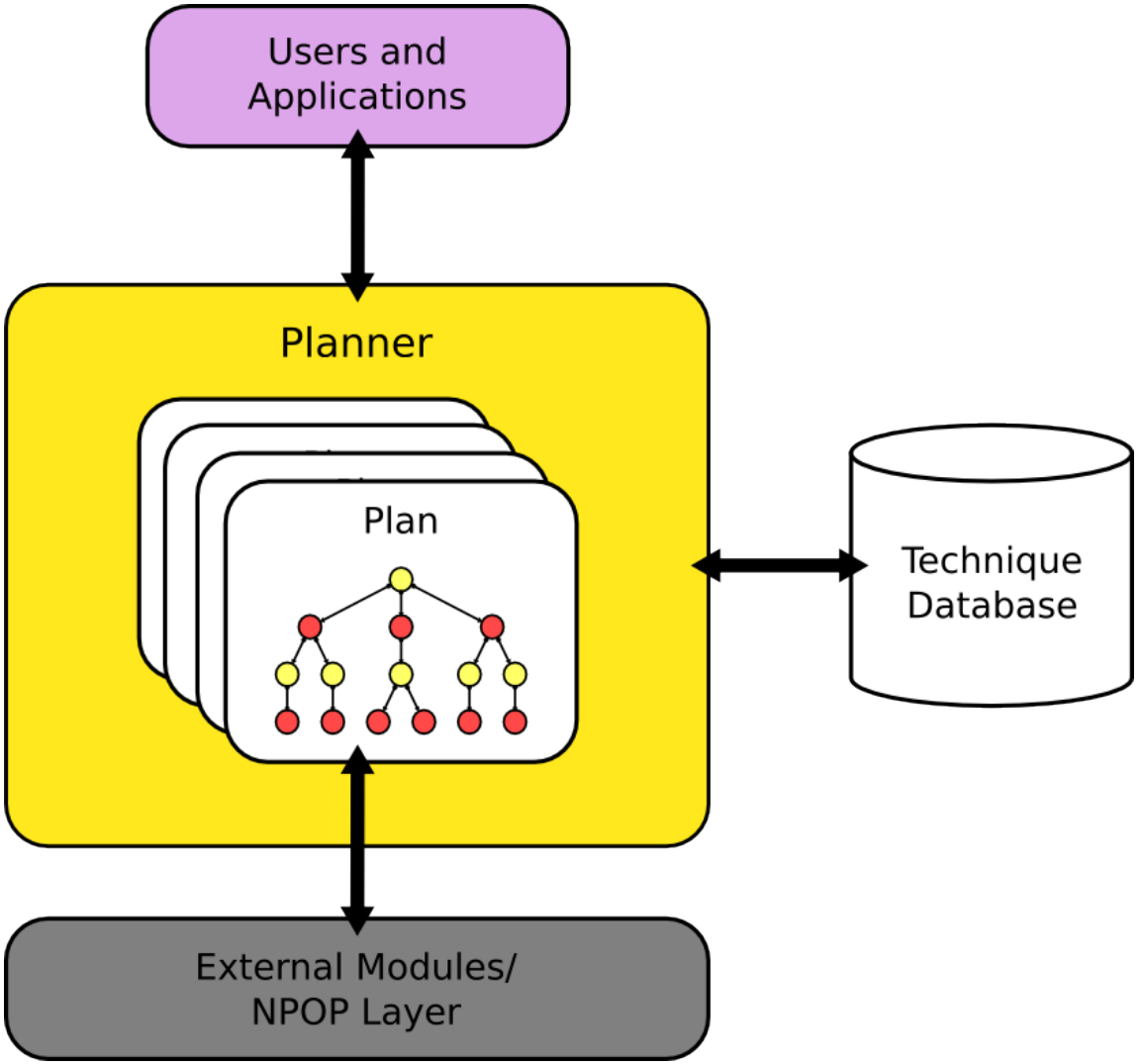
Contents

- Planner Architecture
- Main Concepts
- Invoking the Planner
- Writing Techniques
  - Bridge Techniques
- Understanding Satisfaction

Scope: Planning Technology



Planner Architecture



Planner Architecture

- The Planner sits between users and a lower-level component system.
  - Separable from NPOPs
- Technique Database separate from the Planner
  - Support a variety of Technique Discovery Methods
- A single Planner can run multiple, concurrent Plans

Main Planner Concepts

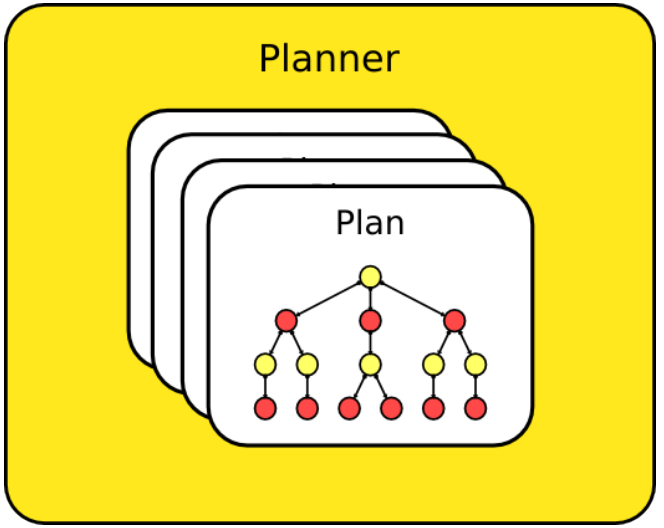
- Goal – a parameterized specification of intent or a desired condition
  - Goal Instance – a Goal with its parameters bound to values.
- Technique – stylized code that implements a Goal
  - Technique Instance – a running object with its parameters bound by a Goal Instance.

Main Planner Concepts

- Solution – description of the service that a Technique can provide
  - “return value” of a sub-goal
  - interface between a Technique and its subgoals (and vice versa)
- Solutions have Properties
  - satisfaction Property used to rank Techniques that can potentially satisfy a Goal
  - all Technique state should be stored in Solution Properties

Main Planner Concepts

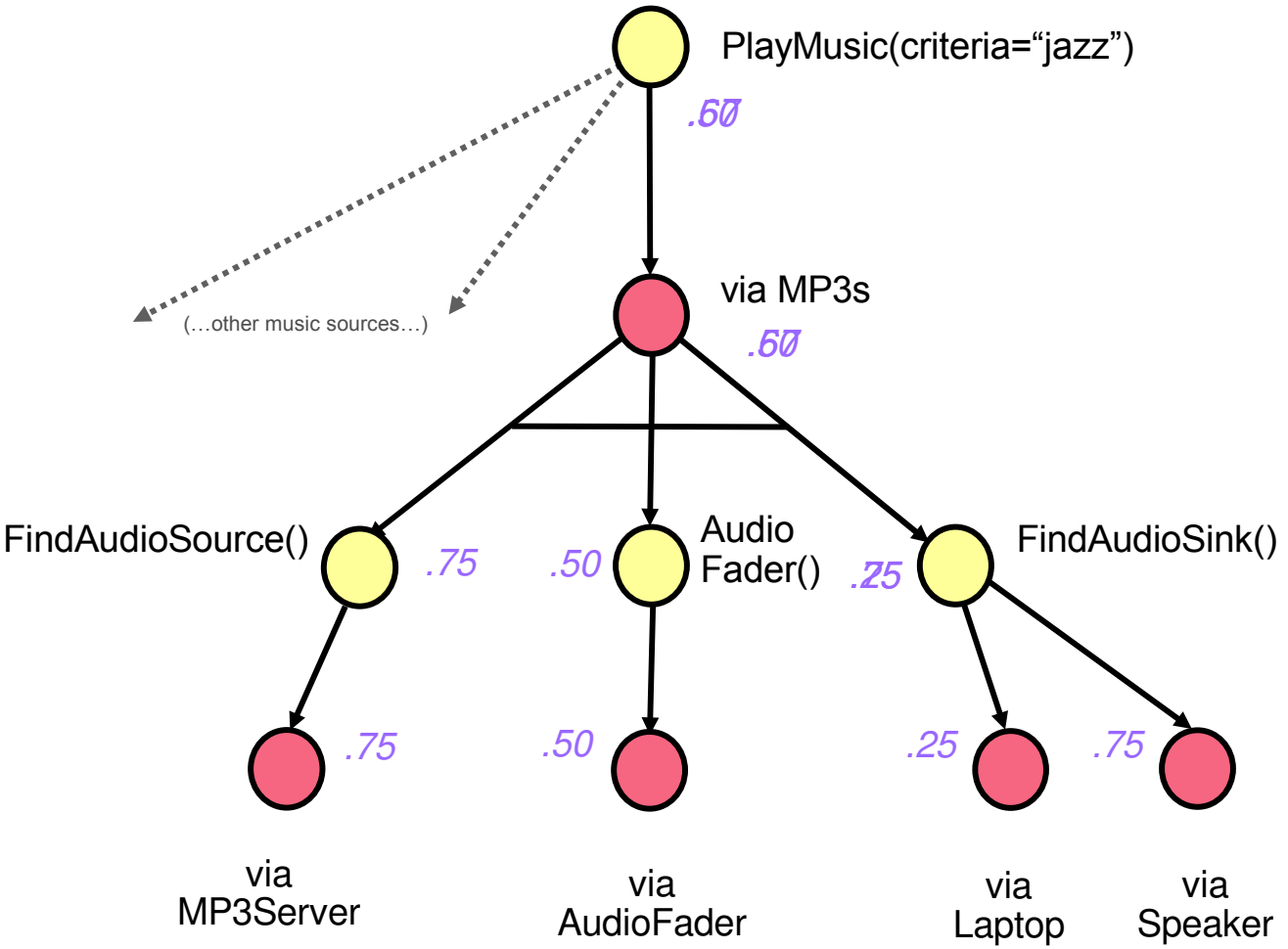
- Planner – runtime system that recursively matches Goals to Techniques.
  - Assert Goals using the **satisfy** call
  - satisfy call returns a Plan – a handle to the Plan Tree associated with the Goal



The Planning Process

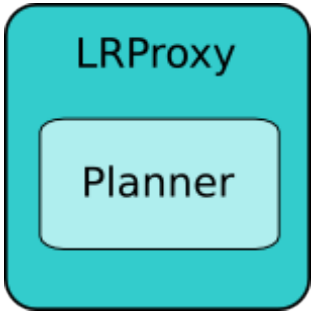
- Build a Plan Tree rooted at top-level Goal
- Evaluate the Plan Tree and choose a suitable Plan.
- Execute the Plan
- Monitor the Techniques in the Plan Tree.
  - Update the executing Plan as necessary.

The Planning Process



Invoking the Planner

- The Planner can be embedded in another application or run at the command line.
  - The embedded Planner API allows more control over the Planning process.



  - Command-line planner useful for testing and legacy applications.

Invoking the Planner

- The Planner, by default, will use a “filesystem” Technique Database
  - Set the TDB_PATH environment variable
  - Also available as the --tdb-path option to the command-line Planner

Invoking the Planner

```
%python planner/planner.py --mode MODE GOAL [...Goal arguments...]
```

- MODE is “once”, “continuous”, or “continuous-update”
- GOAL is the name of the Goal to be run
- The Goal arguments take the form:
  - “-S name=value” for string arguments
  - “-I name=value” for integer arguments
- press “q” then enter to quit

```
%python planner/planner.py --mode until_quit PlayMusic -S criteria=jazz
```

Contents

- Planner Architecture
- Main Concepts
- Invoking the Planner
- Writing Techniques
  - Bridge Techniques
- Understanding Satisfaction

Writing Techniques

- Techniques are the primary way of extending the Planner
  - Write a Technique
  - Put it where the Planner can find it (or set TDB_PATH)
  - JustPlay!
- Techniques are real code – they are processed into Python classes, then executed.

Writing Techniques

```
to PlayMusic(criteria):

    via MP3s:

    subgoals:

        source = FindMusicStream(criteria=goal.criteria)
        sink = FindAudioSink()
        fader = AudioFader()

    eval:

        # Bonus Points for Hot Swapping
        solution.satisfaction = 1.10 * (subgoals.source.satisfaction +
                                        subgoals.sink.satisfaction) / 2

    exec:

        print "Connecting everything together"
        solution.source_out = subgoals.source.resource.get_connector()
        solution.sink_in = subgoals.sink.resource.get_connector()

        subgoals.fader.resource.connect(solution.source_out,
                                        solution.sink_in)

    shutdown:

        print "shutdown via MP3"
        subgoals.fader.resource.replace_sink(solution.sink_in)
```

Programming Model

- Techniques are divided into stages
  - each stage represents the smallest part that the Planner will evaluate at once.
  - Two kinds:
    - Evaluation Stages
    - Execution Stages
- Write stage N as though all previous stages have
- Store all state in your Solution object
  - available as the “solution” variable

Technique Evaluation Stages

- Evaluation stages refine the Properties of the Technique's Solution.
  - Make sure that you have non-zero satisfaction at all time, unless you've failed.
  - Properties other than satisfaction may be set.
- Evaluation stages must be idempotent
  - The Planner keeps track of what Properties each stage in the Technique uses. If a Property changes, that stage will be re-run.
- Store all state in your Solution!

Technique Evaluation Stages

- first: run some basic initialization
- subgoals: declare subgoals
  - sg = GoalName(args)
  - Solution object of “sg” available in all later stages in “subgoals.sg” variable.
- eval: run Python code to evaluate the environment
  - (technically) time limited
- foreground: same as eval, but not time limited and does not block the planner.

Technique Execution Stages

- exec: Python code to execute if this Technique is chosen
- shutdown: Python code to execute when shutting down this Technique
- update *sg* from *var*: Python code that will hotswap the subgoal named “sg”.
  - new Solution available as “subgoals.sg”
  - old Solution available as “subgoals.var”

Technique Stages

```
to PlayMusic(criteria,preferred_artists):
    via PreferMyArtists:

    subgoals:
        source = FindMusicStream(criteria=goal.criteria)
        sink = FindAudioSink()

    eval:

        solution.satisfaction = (subgoals.source.satisfaction +
                                 subgoals.sink.satisfaction) / 2

    eval:
        if subgoals.source.artist in goal.preferred_artists:
            solution.satisfaction = 1.10 * solution.satisfaction

    exec:
        solution.source_out = subgoals.source.resource.get_connector()
        solution.sink_in = subgoals.sink.resource.get_connector()

        Connect(solution.source_out,
                solution.sink_in)

    shutdown:

        print "shutdown via MP3"
        subgoals.fader.resource.replace_sink(solution.sink_in)
```

source

satisfaction: .5 .7

artist: Ella F... Miles D...

sink

satisfaction: .5

this.solution

satisfaction: .5 .55 .6

Technique Tips

- Store all state in solution objects!
- Use subgoals as much as possible.
  - Increases adaptivity
  - Allows more customization

Bridge Techniques

- Low-level Techniques must discover and interact with devices.
  - low-level code shouldn't be in a Technique,
  - but must interoperate with the Technique
- Avoid ad-hoc decision making.
- GenerateSolution
  - “Built-in” subgoal that can call long-running code

GenerateSolution

- GenerateSolution(code, _shutdown, *args, **kwargs)
  - code: a function to call
  - _shutdown: how to shutdown the function
- Code run from GenerateSolution is supplied with a “SolutionFactory”
  - Produces solutions
  - Causes the original Technique to be cloned

GenerateSolution

- JustPlay uses GenerateSolution to interact with the Registry
  - Every new Entity is represented by a new Solution object
  - See playground/audiohotswap/finders.teq for examples

Using New Techniques

- Add them to your Technique Database directory...
  - (or set TDB_PATH)
- Restart your Planner

Contents

- Planner Architecture
- Main Concepts
- Invoking the Planner
- Writing Techniques
  - Bridge Techniques
- Understanding Satisfaction

Satisfaction

- Satisfaction is a measure of how well a Technique satisfies its Goal
  - 0 to 1 – 0 is complete failure, 1 is perfection
- Allows distributed evaluation
- Is a reduction from Properties (rich-valued multi-valued descriptions) to a scalar
  - Loss of information

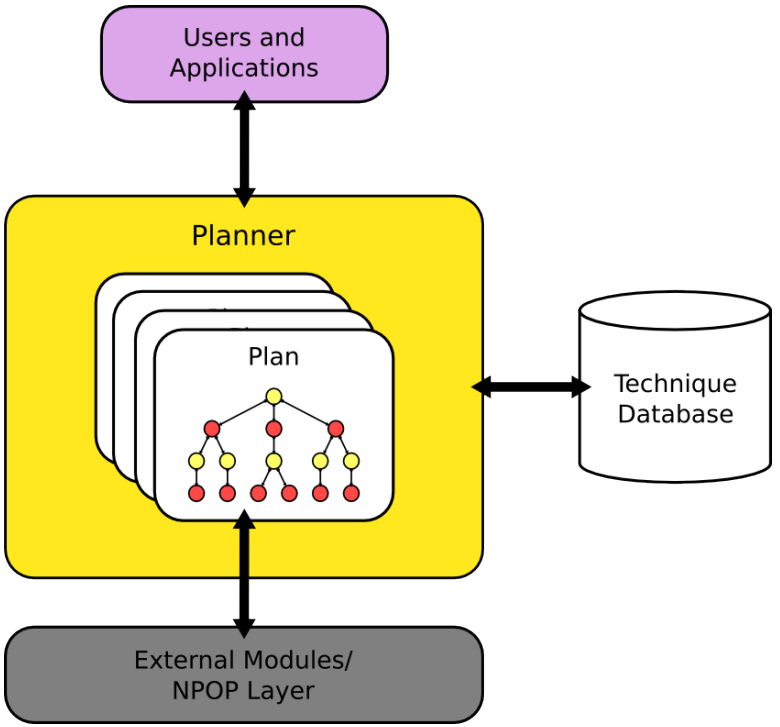
Satisfaction

- Techniques are required to set a satisfaction property.
  - Satisfaction values of subgoals
  - Information from functions in eval/foreground stages
- The Planner must trust the satisfaction calculation by Techniques
  - The Planner can learn about “liars”

Contents

- Planner Architecture
- Main Concepts
- Invoking the Planner
- Writing Techniques
  - Bridge Techniques
- Understanding Satisfaction

Lab Session



Day 2, Session 2
July 2006
Quanta JustPlay Seminar

Justin Mazzola Paluska
jmp@mit.edu