Progress in Natural Language and Knowledge Representation

David Devault (ICT), Arno Hartholt (ICT), Eduard Hovy (ISI), Anton Leuski (ICT), Tom Russ (ISI), David Traum (ICT) | 9/24/2008

(presented by Eduard Hovy)

USC

The projects or efforts depicted were or are sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM),and/or the US Army Research Institute. The content or information presented does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

ICT

INSTITUTE FOR CREATIVE TECHNOLOGIES

Even VHUMANs mature…

- **MRE and SASO: We have built a very sophisticated VHUMAN:**
  - Most innovative design and most complex internal structure in the research literature
  - Powerful capabilities of each module plus rich internal interconnections provide system's flexible and reactive behavior
- **But: the system has been experimental and exploratory**

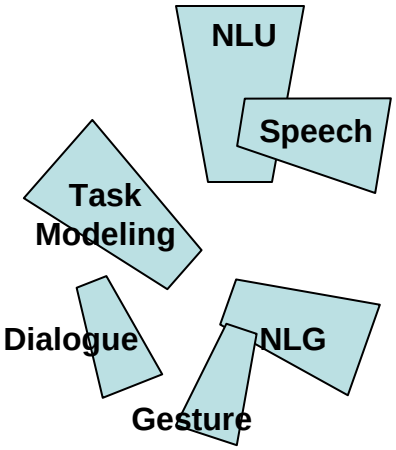
- **Recently, our work has started to mature…**
  - We want to be able to **extend** the system (e.g., add new agents) **rapidly**
  - We want to **create new scenarios** easily
  - We would like to **enable other people** to build scenarios
  - Many of our **modules** are beginning to be **used in other projects**

- **This is typical R&D evolution: idea → pilot → prototype → re-use → distribution**

- **So… We are re-engineering and centralizing much of the infrastructure to make cross-module growth and changes easy and consistent**

Current state of affairs

- **Problem: A Babel inside the system:**
  - *N* major modules … each performing a different function
  - BUT each has its own different internal representation terms and modeling style
  - Why?
    - We started with some advanced legacy systems — building everything fresh would have taken too long
    - Each module is pushing the state of the art in its own research field — there's no reason their representations and methods should correspond



- **This makes it a LOT of work to extend functionality that crosses modules:**
  - For example, you can add new words to NLU, but that doesn't mean that NLG knows the words!

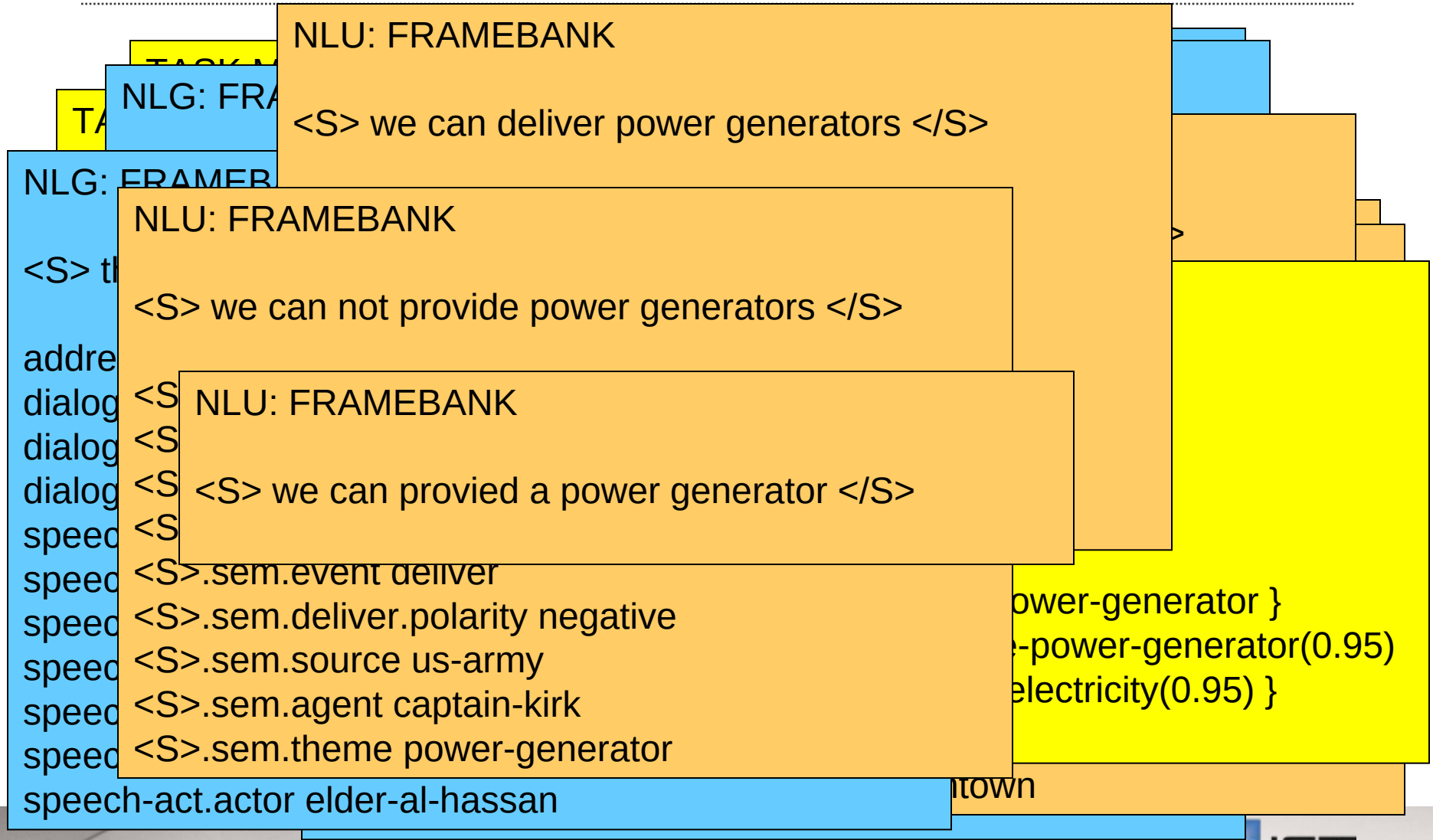
- **Desiderata for 'one-shot extensibility'**
  - Should be easy to build new knowledge
  - New additions should be made consistent, also with existing knowledge
  - Knowledge should be compositional: new knowledge should plug into older knowledge
  - Knowledge should be used by all modules in the same ways

USC
ICT
INSTITUTE FOR CREATIVE TECHNOLOGIES

Example: Extending the system

- **Want to add new line of negotiation:**
  - EXISTING: Doctor can negotiate for delivery of medical supplies downtown
  - NEW: Want Elder to negotiate delivery of power generator downtown
  - So: Elder needs to know
    - Task model module: Define “deliver” action, its preconditions, effects, dependencies on other entities:
      - Operating the clinic downtown requires power
      - Having power downtown would be a good thing
      - Precondition for power is a power generator
      - There isn’t one today, so it has to be obtained and delivered
      - etc.
    - NLU and NLG and Speech modules:
      - Lexicons: Words for “power generator”, “deliver”, “need”, etc.
      - Grammars: Patterns and phases using these words that can accommodate the negotiation
      - Semantics: Representation frames that represent these concepts
- **Currently this is hard to do: even though the Doctor knows all this, the Elder doesn’t, and his various modules must *each* be extended**

Example: Adding new knowledge (old method)



Example: Adding knowledge for NLU

- **Of course we create tools and interfaces to speed up system extension**
- **Still, it can be a lot of work to add the knowledge needed for NLU:**
  - Define the words
  - Define the representation frame elements
  - Link them together
  - Ensure that they integrate with everything else

QuickTime™ and a
H.264 decompressor
are needed to see this picture.

Today: Extending NLG, using auto-generation

- **Evolved specialized procedure to facilitate extensions:**
  - Builder adds knowledge for one new phrase
  - System automatically creates variations, using grammar
- **For NLG, sequence is:**
  - Want to say “power generator”
  - Generator can’t produce sentence
  - Addition interface: provide needed knowledge (yes, it’s ugly!)
  - Back to NLG: now produces sentence
  - …AND: its auto-extension produces six more variations for free!
  - BUT: this is not yet centralized, not exported to NLU…

Extending NLG: Before addition

QuickTime™ and a
Video decompressor
are needed to see this picture.

Extending NLG: Making addition

QuickTime™ and a
Video decompressor
are needed to see this picture.

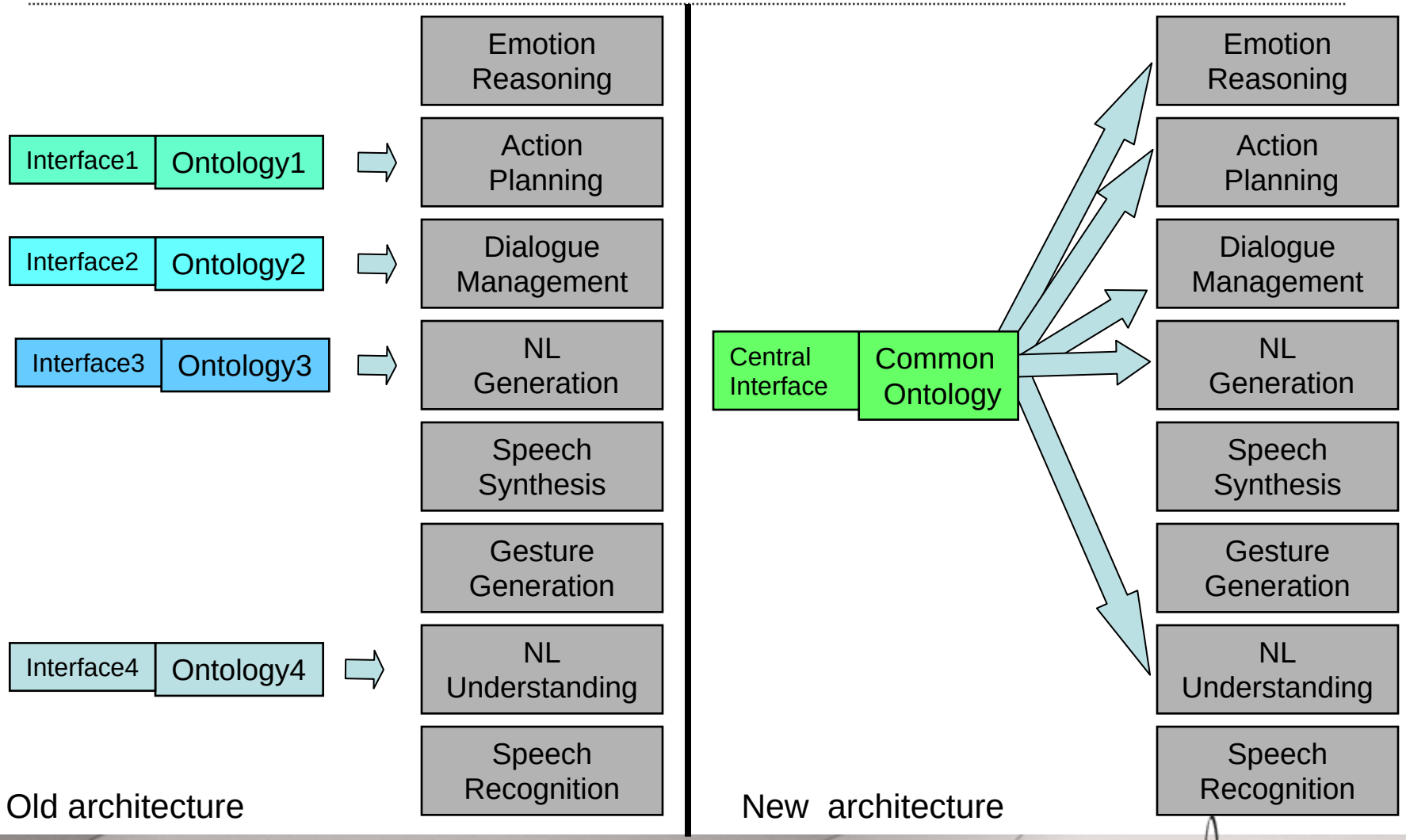
Extending NLG: After addition

QuickTime™ and a
Video decompressor
are needed to see this picture.

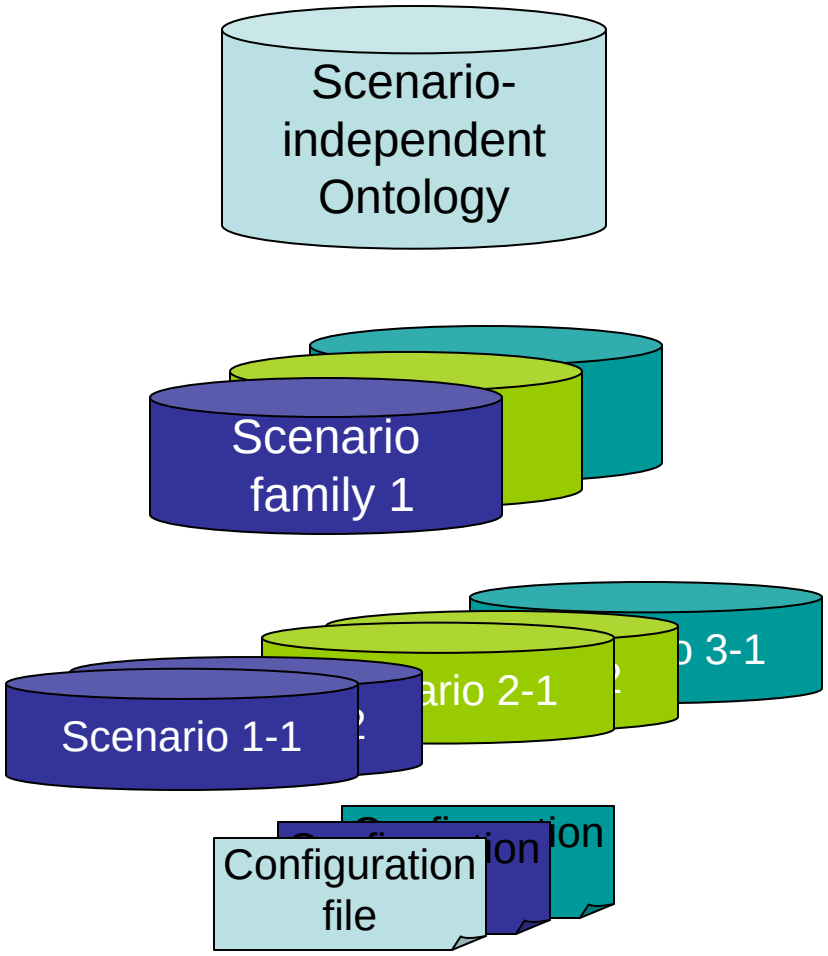
Where are we going? A centralized framework

- **A single central ontology of terms:**
  - Standardizes all the terms used by all modules
  - Makes explicit most of the detailed knowledge currently held by experts
- **A single content-building/checking interface (Protégé, from Stanford):**
  - Provides a single point of entry of all necessary info, all related
  - Supports a scripted series of content-building steps, during which the interface requests different related kinds of info (as needed for each module)
- **Automated internal consistency checking**
- **Automated content enhancer/exporter functions:**
  - Automatically inherits required info from ontology and fleshes out new input material
  - Automatically converts fleshed-out input into formats used by various modules

Old and new architecture



New ontology organization



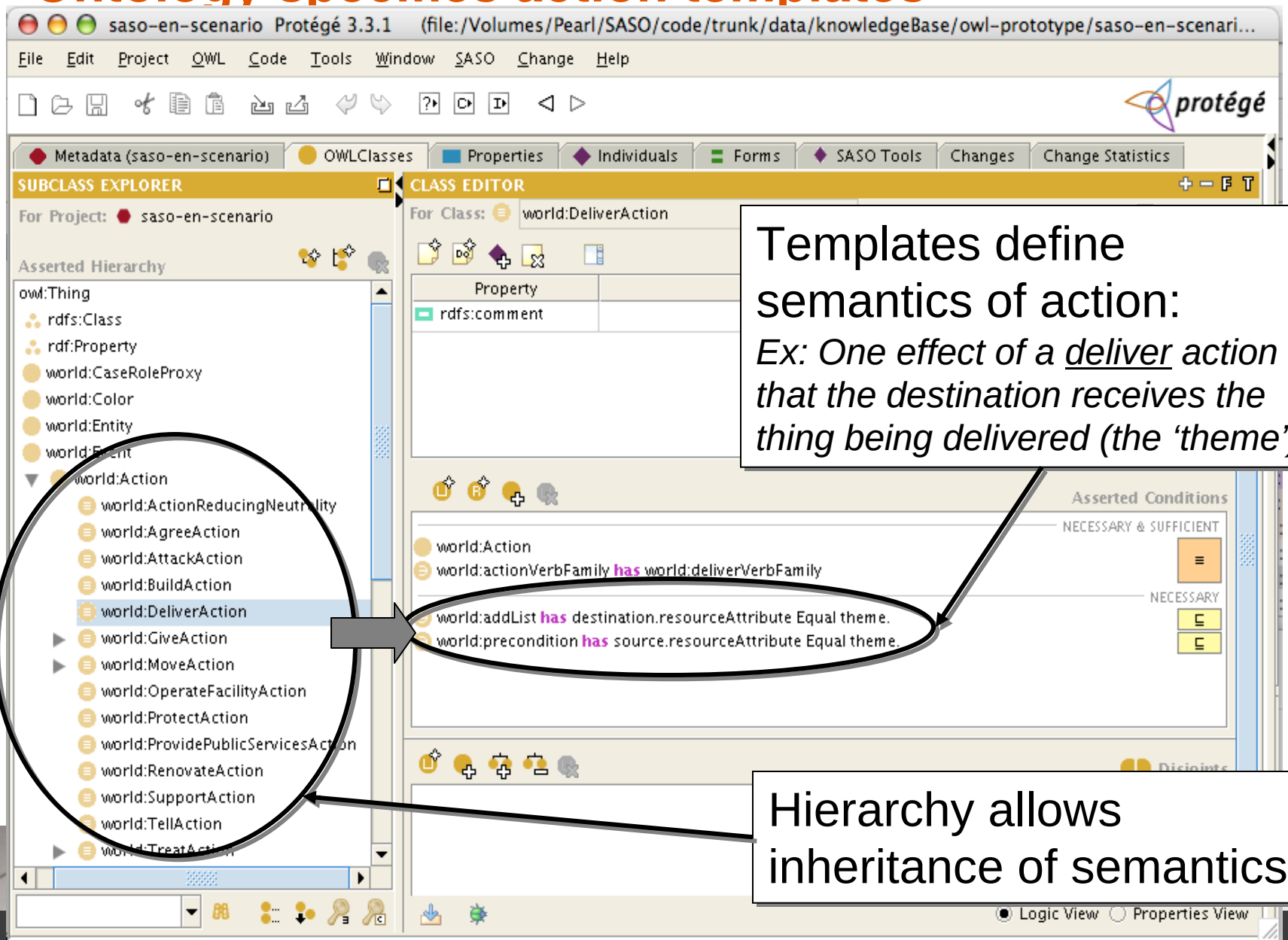
- General world knowledge
  - (generic actions, objects…)
- Linguistic structures
- Generic dialogue items

- Scenario actor classes
- Scenario action templates
  - (generic preconds, effects...)
- Propositions for scenario

- Specific actors, locations, etc.
- Sentences for scenario

- Actor positions, attitudes, etc.
- Simulation initialization

Ontology specifies action templates

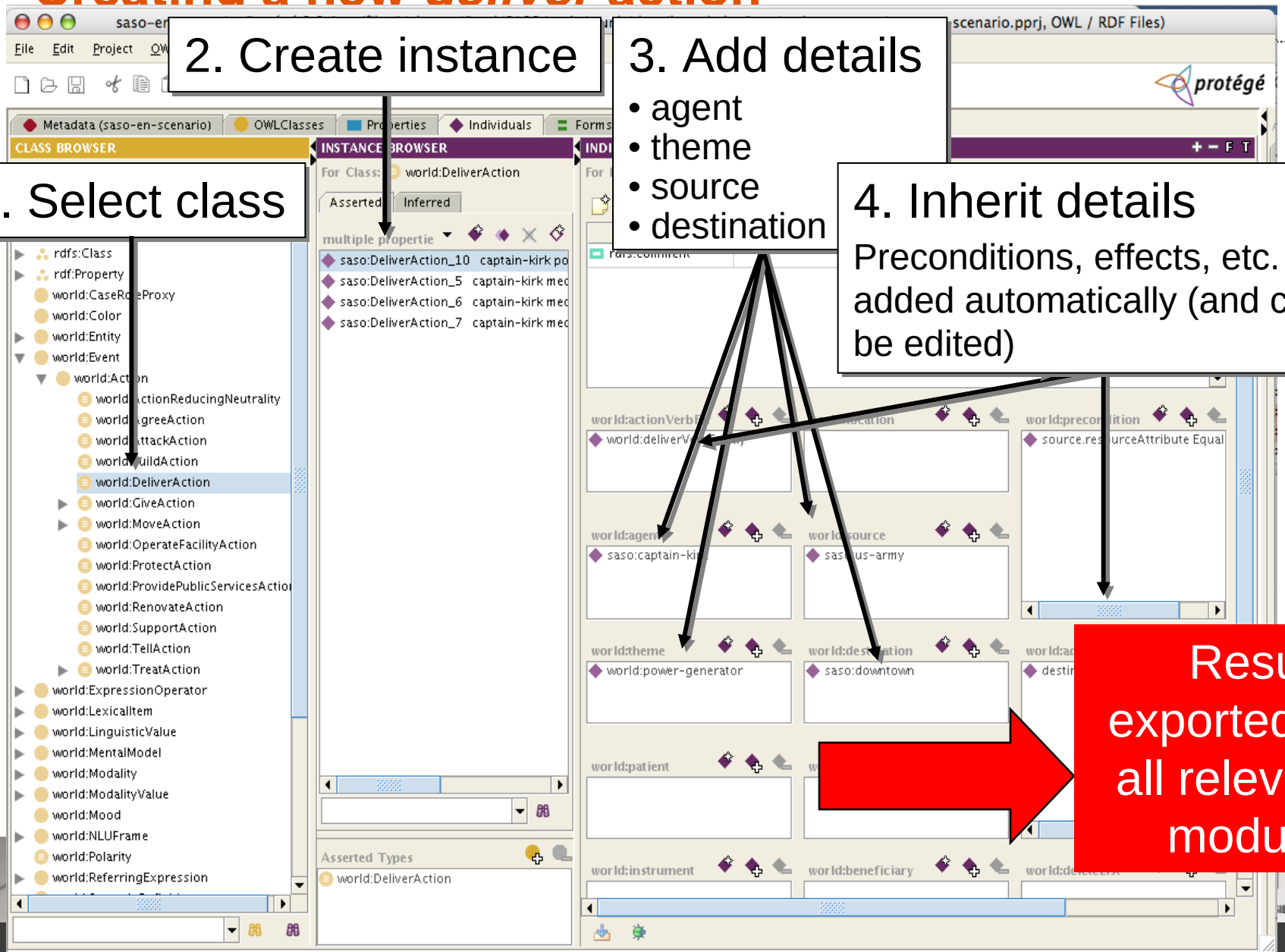


Templates define semantics of action:

Ex: One effect of a deliver action is that the destination receives the thing being delivered (the 'theme')

Hierarchy allows inheritance of semantics

Creating a new *deliver* action



Templates simplify adding *deliver*

Old method

Create new Deliver action instance
Specify agent = “CaptainKirk”
Specify theme = “power-generator”
Specify source = “us-army”
Specify destination = “downtown”

Create state “downtown-has-power-generator”
Create task model “downtown-has-power-generator”

Specify belief
Specify initial-value
Create state “us-army-has-power-generator”
Create task model “us-army-has-power-generator”
Specify belief
Specify probability
Specify initial-value

Create new task instance
Specify ground event is the deliver action instance
Specify precondition “us-army-has-power-generator”
Specify add-effect “downtown-has-power-generator”

New centralized method

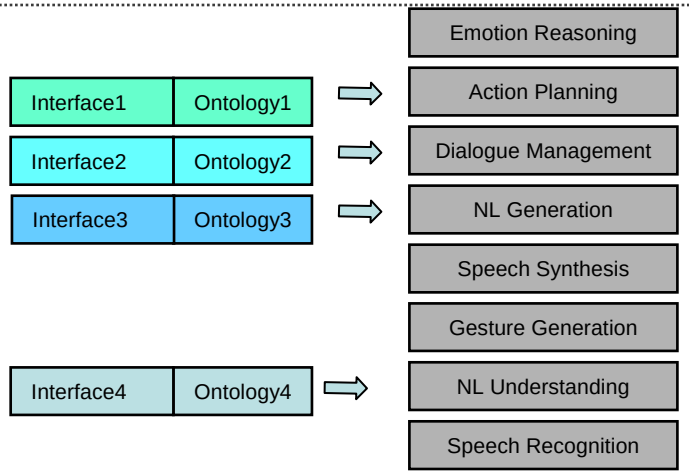
Create new Deliver action instance
Specify agent = “CaptainKirk”
Specify theme = “power-generator”
Specify source = “us-army”
Specify destination = “downtown”

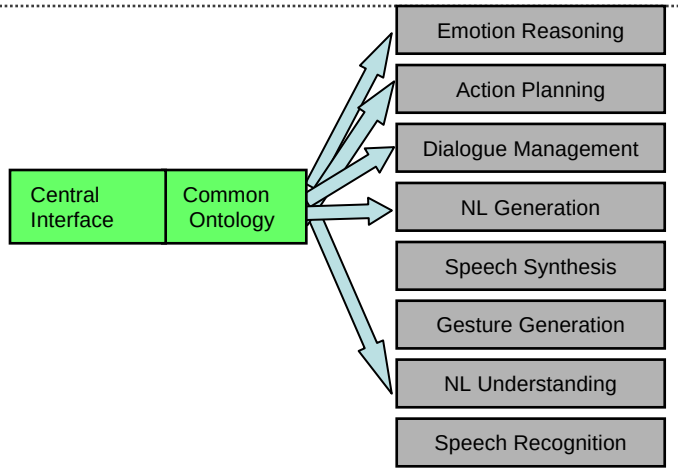
Create state “downtown-has-power-generator”
Create task model state
“downtown-has-power-generator”
Specify belief

Create new task instance
Specify ground event is the deliver action instance

USC
ICT
INSTITUTE FOR CREATIVE TECHNOLOGIES

Work required: The old and the new





- **In the past: system builder**
  - identify core problem,
  - locate problem in each module,
  - design all fix/extension(s),
  - implement change(s) in each module,
  - coordinate change across modules,
  - perform extensive tests,
  - perform cross-module debugging.

- **In the future: system builder**
  - identify core problem,
  - design fix/extension,
  - implement change in central repository, letting tools perform consistency check
  - perform final tests and one-shot centralized debugging.

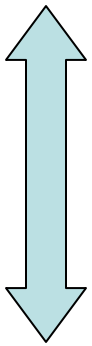
Where next?

- **We're partway through the work…**
  - Completed central standard ontology
    - Using Stanford's Protégé for content building
    - Using Protégé's interface for ontology access/browsing
    - Using OWL (Semantic Web) representation formalism
    - Used some concepts from various upper-level ontologies, such as SUMO and Omega
  - Completed exporter functions to some modules
- **To be done in coming year:**
  - Integrating other modules (incl. additional exporter functions)
  - Completing consistency and integrity testing routines
  - Completing authoring environment: Extending interfaces to support specialized authoring
  - Exploring software 'props warehouses' for new scenarios:
    - Repository of objects, standard actions, desires, plans, locations, etc.
    - Everything connected to ontology

Research contribution

- **Integrated ontology-driven extensibility facilitates:**
  - Rapid authoring of new scenarios, new capabilities, new agents
  - Reduction of error, making expert knowledge explicit, etc. --- Integration, consistency checking, etc.
  - Flexible planning and scripting of training alternatives
- **But…this work is not just plumbing!**

- **We want to enable non-CS people to author new scenarios…so we're investigating the optimal point in the tradeoff between**
  - Programming for Non-Experts:
    - Reduced, simple, scripting languages
    - BUT limited in functionality
    - Ex.: Sgt Blackwell internal details
  - Powertools for Experts:
    - High functionality
    - BUT need considerable expertise
    - Ex.: VHUMAN's TCL, Soar, and other components



Simpler, but limited

More powerful, but complex

USC

ICT INSTITUTE FOR CREATIVE TECHNOLOGIES

Thank you