CRE Omega Demo: Progress Report and Technical Documentation

Common Runtime Environment (CRE) Project

February 2026

Contents

1 Project Overview

1.1 CRE: Common Runtime Environment

CRE is an Erlang/OTP project implementing a YAWL (Yet Another Workflow Language) workflow engine with Petri Net-based patterns. Key characteristics:

- **Language**: Erlang/OTP 25+ (tested through OTP 28)
- **Build**: rebar3
- **Core dependency**: `gen_pnet` (the only OTP process maintaining state)
- **Design**: Joe Armstrong style—one real OTP runner, everything else pure helpers + message contracts

1.2 Architecture

- **Single runtime component**: `gen_pnet` maintains workflow state
- **Pure helpers**: `pnet_types`, `pnet_marking`, `pnet_mode`, `yawl_compile`, `yawl_pattern_expander`, `wf_yawl_executor`
- **Key modules**: `yawl_engine`, `yawl_executor`, `cre_yawl_patterns`, `gen_yawl`

2 AGI Symposium Omega Demo

2.1 Purpose

The Omega demo executes the full AGI Symposium workflow with:

- **20 LLM agents** simulating human committee roles (Chair, Program Chair, Reviewer, Ops Lead, Venue Lead, etc.)
- **43 YAWL workflow patterns** (P1–P43) composed in a single specification
- **Swarm Turing Test**: Can the system produce a transcript indistinguishable from human committee deliberation?

2.2 Entry Points

Listing 1: Omega demo commands

```
./scripts/run_agi_symposium_demo.sh --omega            # With Z.AI (
    ZAI_API_KEY required)
./scripts/run_agi_symposium_demo.sh --omega --dry-run # Curated
    responses, no LLM
```

2.3 Specification

The workflow is defined in `test/fixtures/agi_symposium_omega.yaml`:

- Root net: `Symposium`
- Subnets: `ProgramThread`, `OpsThread`, `CommsThread`, `IncidentThread`, `SatelliteSymposium`
- Patterns: P42 ThreadSplit (split), P41 ThreadMerge (merge), P3 Synchronization (GoNoGo), P38 GeneralSyncMerge (CloseSymposium), plus 39 others

3 Problem: Blocking at Zero Rounds

3.1 Observed Behavior

Prior to the fix, running `./scripts/run_agi_symposium_demo.sh -omega` produced:

- `Final State: blocked`
- `Rounds: 0`
- No transitions fired; workflow never started

3.2 Root Cause: Pattern Collision

The Symposium root net has **13+ pattern instances** that expand into Petri net structures. When merged via `merge_net_structures`, internal transition and place names **collided**:

Table 1: Transition name collisions across patterns

Pattern	Transitions	User-facing rename
P42 ThreadSplit	`t_split`, `t_finish1..4`	`t_split` → `t_SplitMegaThreads`
P41 ThreadMerge	`t_split`, `t_complete1..4`, `t_merge`, `t_finish`	`t_merge` → `t_MergeMegaThreads`
P3 Synchronization	`t_split`, `t_complete1..3`, `t_join`, `t_finish`	`t_join` → `t_GoNoGo`
P38 GeneralSyncMerge	`t_split`, `t_complete1..3`, `t_join`, `t_finish`	`t_join` → `t_CloseSymposium`

`maps:merge` uses last-writer-wins. The final preset/postset for `t_split` came from whichever pattern was processed last, producing a structurally broken net. In the worst case, a transition consumed the initial token but produced nothing (`#{}`), causing immediate deadlock.

3.3 Place Collisions

Shared place names (`p_start`, `p_end`, `p_joined`, `p_branch1..3`) also collided. P3 and P38 both use `p_branch1..3`; the place mapping should produce `p_gonogo_branch1..3` and `p_close_branch1..3`, but one overwrote the other.

4 Solution: Per-Instance Namespacing

4.1 Strategy

Namespace all **internal** (non-renamed) transitions and places per pattern instance. User-facing transitions (`t_GoNoGo`, `t_SplitMegaThreads`) keep their names.

4.2 Implementation

4.2.1 Transition Namespacing

`add_internal_transition_namespace/3` in `yawl_pattern_expander.erl`:

- For each transition **not** in `TrsnMap` keys, add mapping: `t_split` → `t_pi<N>_split`
- Prefix `pi<N>` derived from `erlang:phash2(IdStr)` for pattern instance id

4.2.2 Place Namespacing

`add_internal_place_namespace/4`:

- For each place **not** in `PlaceMap` keys, add mapping: `p_branch1` → `p_pi<N>_branch1`

- **Exception**: `p_start` for the **entry-owning** pattern stays unnamespaced so `init_marking` finds it

4.2.3 Entry Owner Detection

`compute_entry_owner_id/3` in `yawl_compile.erl`:

- Uses `wf_yaml_spec:net_first_flow_target/2` to get first flow target from input condition
- For Symposium: `Start` → `SplitMegaThreads`
- Pattern instance with `split_task: SplitMegaThreads` (P42) is the entry owner

4.2.4 Subnet Wiring Fix

`omega_demo_runner` expects `p_gonogo_branch1..3` for P3 sync. Added `join_task_to_suffix(GoNoGo)` → `«"gonogo"»` so P3 produces `p_gonogo_branch1..3` instead of `p_GoNoGo_branch1..3`.

5 Files Modified

Table 2: Summary of code changes

File	Changes
`src/core/yawl_pattern_expander.erl`	`add_internal_transition_namespace`, `add_internal_place_nam`
`src/core/yawl_compile.erl`	`compute_entry_owner_id`, `pattern_net_matches`, pass `entry_own`
`src/wf/wf_yaml_spec.erl`	`net_first_flow_target/2`, `input_cond_to_atom/1`

6 Current Progress

6.1 Results

Table 3: Omega demo execution before and after fix

Metric	Before	After
Rounds	0	3
Final State	blocked	blocked
Initial token consumed	No	Yes
Transitions fired	0	≥ 3

6.2 Namespacing Verification

The compiled `yawl_Symposium` module now contains:

- `t_pi36722154_complete1..3`, `t_pi103154468_complete1..3` (namespaced internal transitions)
- `p_gonogo_branch1..3`, `p_close_branch1..3` (correct P3/P38 place names)
- `p_pi29137630_branch1..3` (namespaced internal places)

6.3 Remaining Issue: Marking Cycle

The demo logs: `gen_yawl halt: marking cycle detected (fingerprint=57720917)`. This indicates:

- The workflow reaches a marking that has been seen before (cycle detection in `gen_yawl` continue handler)
- Execution halts to prevent infinite loops
- "Blocked" after 3 rounds is due to cycle detection, not necessarily absence of human tasks

7 Thesis Interpretation: What Three Rounds Mean

7.1 Minimal Witness of Compositional Execution

Three rounds represent the **first non-trivial witness** that the workflow engine executes composed patterns correctly:

1. **Round 0**: No execution; specification structurally broken
2. **Rounds 1–3**: Automated splits (P42), subnet runs (ProgramThread, OpsThread, CommsThread), and/or merge transitions fire
3. **Full run**: All 43 patterns exercised; Swarm Turing Test potentially passable

7.2 Epistemic Significance

The fix (namespacing) was not "adding features" but **making the specification's semantics consistent with its intended execution**. Three rounds demonstrate that:

- The compiled net is executable
- Multiple patterns can coexist on the same net without semantic collision
- The boundary between "broken" and "partially working" has been crossed

7.3 Swarm Turing Test

The Swarm Turing Test asks: can the system produce a transcript indistinguishable from human committee deliberation? Passing it would demonstrate **framework insufficiency**—that fixed policy spaces cannot express certain workflow-induced behaviors. Three rounds do not pass the test but establish that the engine can **run** the specification far enough to make the question meaningful.

8 Commands and Verification

8.1 Run Omega Demo

```
# With Z.AI (requires ZAI_API_KEY)
ZAI_API_KEY=your_key ./scripts/run_agi_symposium_demo.sh --omega

# Dry run (curated responses, no LLM)
./scripts/run_agi_symposium_demo.sh --omega --dry-run
```

8.2 Compile Omega to Files

```
erl -pa _build/test/lib/cre/ebin _build/default/lib/*/ebin \
    -pa _build/test/lib/yamerl/ebin -noshell -eval \
  '{ok,S}=wf_yaml_spec:from_yaml_file("test/fixtures/agi_symposium_omega
     .yaml"),
␣␣␣{ok,_}=yawl_compile:compile_to_file(S,#{},"/tmp/omega_compiled_dump")
   ,halt(0).'
```

8.3 Test Suites

```
rebar3 eunit --module=yawl_compile   # 13 tests, 0 failures
rebar3 ct --suite=agi_symposium_simulation_SUITE
```

9 Summary

- **Problem**: Omega demo blocked at 0 rounds due to pattern transition/place name collisions when merging 13+ pattern instances on the Symposium net
- **Solution**: Per-instance namespacing of internal transitions and places; entry owner keeps `p_start`; `p_gonogo_branch1..3` wiring fix for `omega_demo_runner`
- **Outcome**: 0 → 3 rounds; workflow is executable; marking cycle detection triggers before full completion
- **Next**: Investigate cycle topology and cycle-detection policy for full Omega run