Erlesy

Visualizing OTP State Machines

Nicholas Gunder ()

and Pawel Antemijczuk ()

Why would anyone do this?

Two Things

- Not everyone will become an Erlang coder

AND

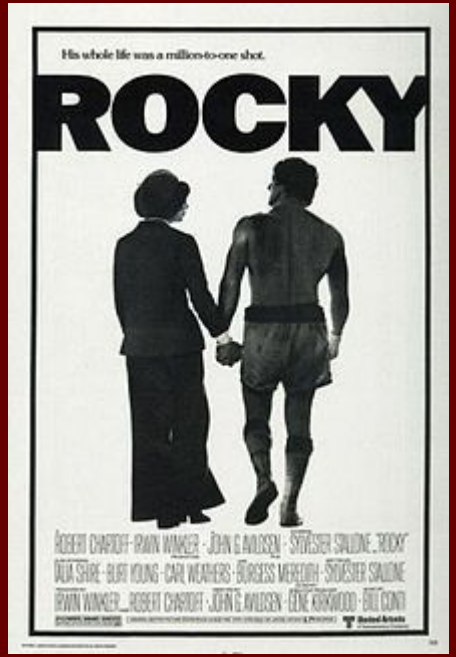
- Good Design is difficult when you have no idea what you should do

Graphs provide

- Quick overview and feedback
- Common understanding
- Iterative design
- Reduced effort in capturing requirements

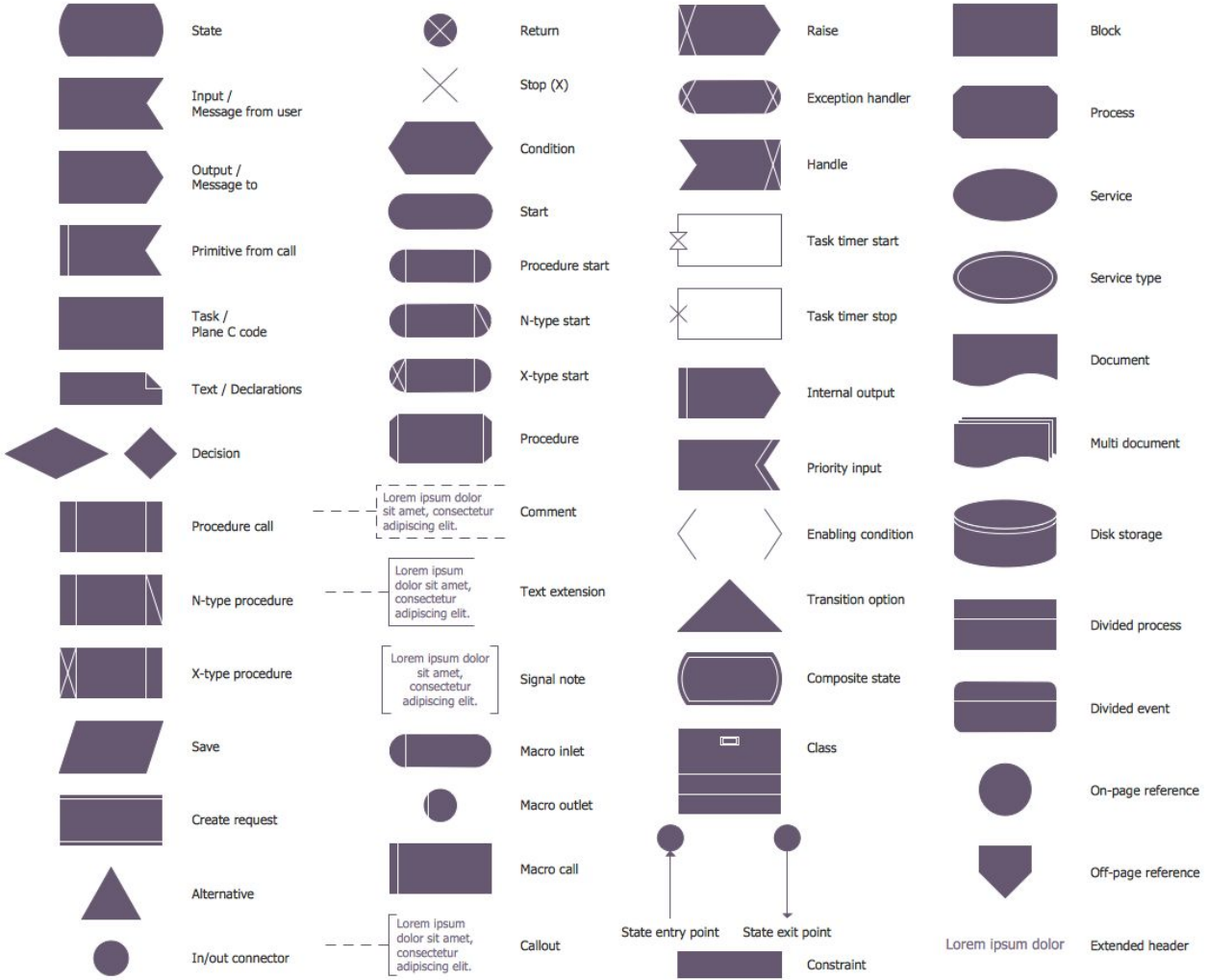
Graphs suck when

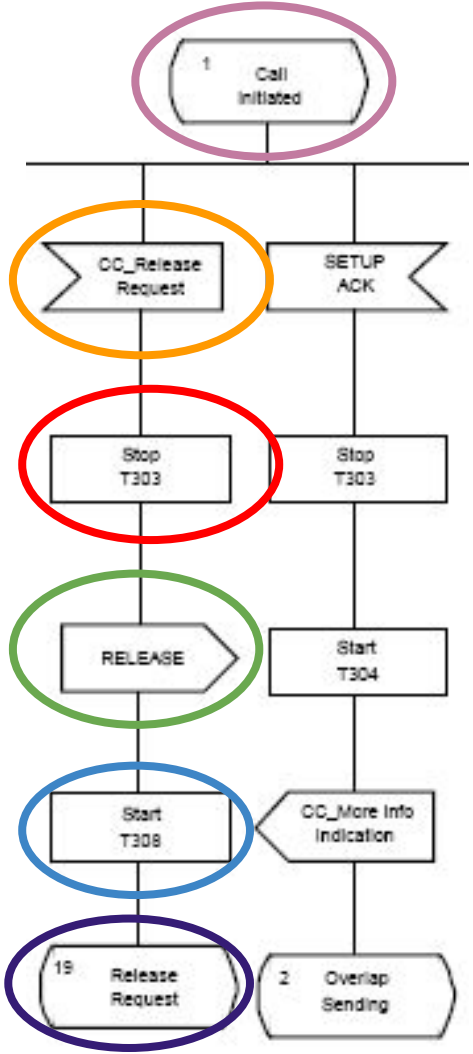
- No one wants to maintain them!
- Structure is not enforced
- People have different opinions on how it should be done
- Code cannot be mapped easily
- They become detached from the implementation



SDL/GR (introduced 1976)

SDL Diagrams

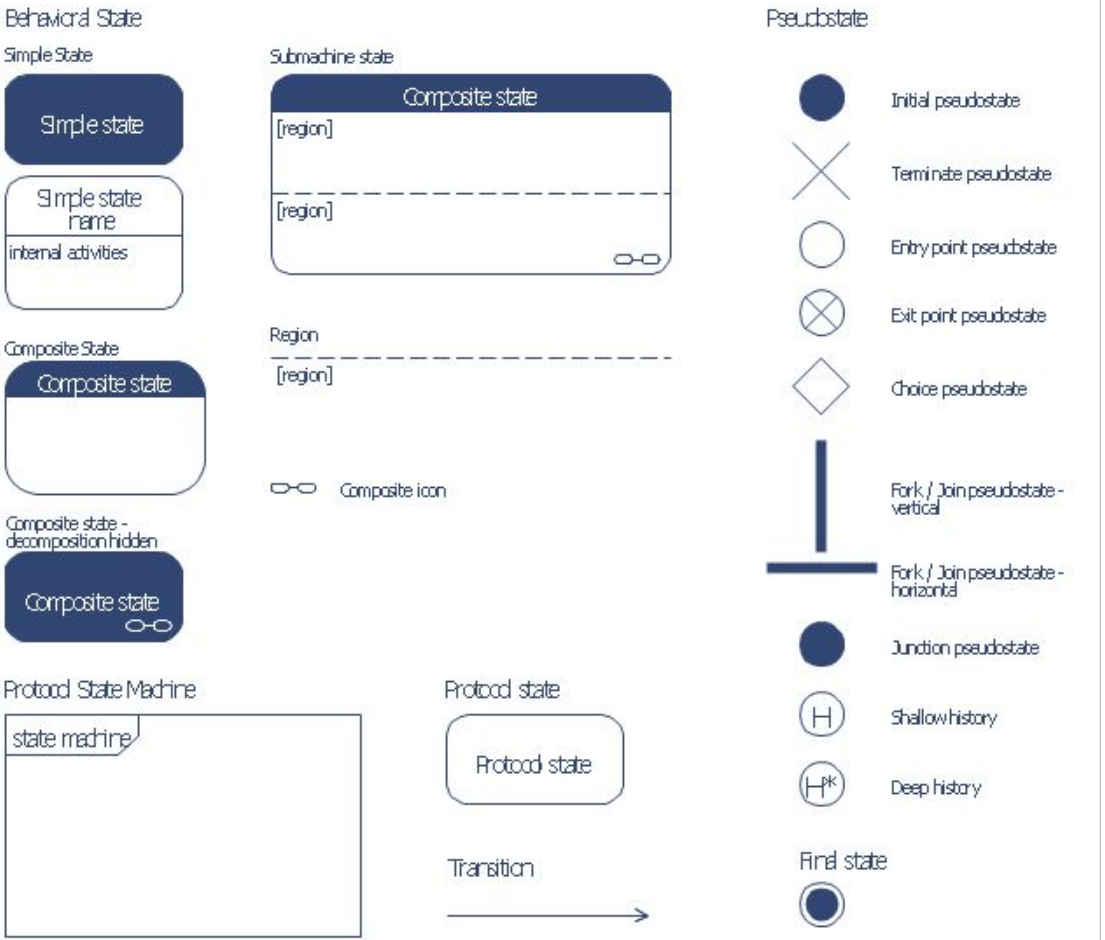


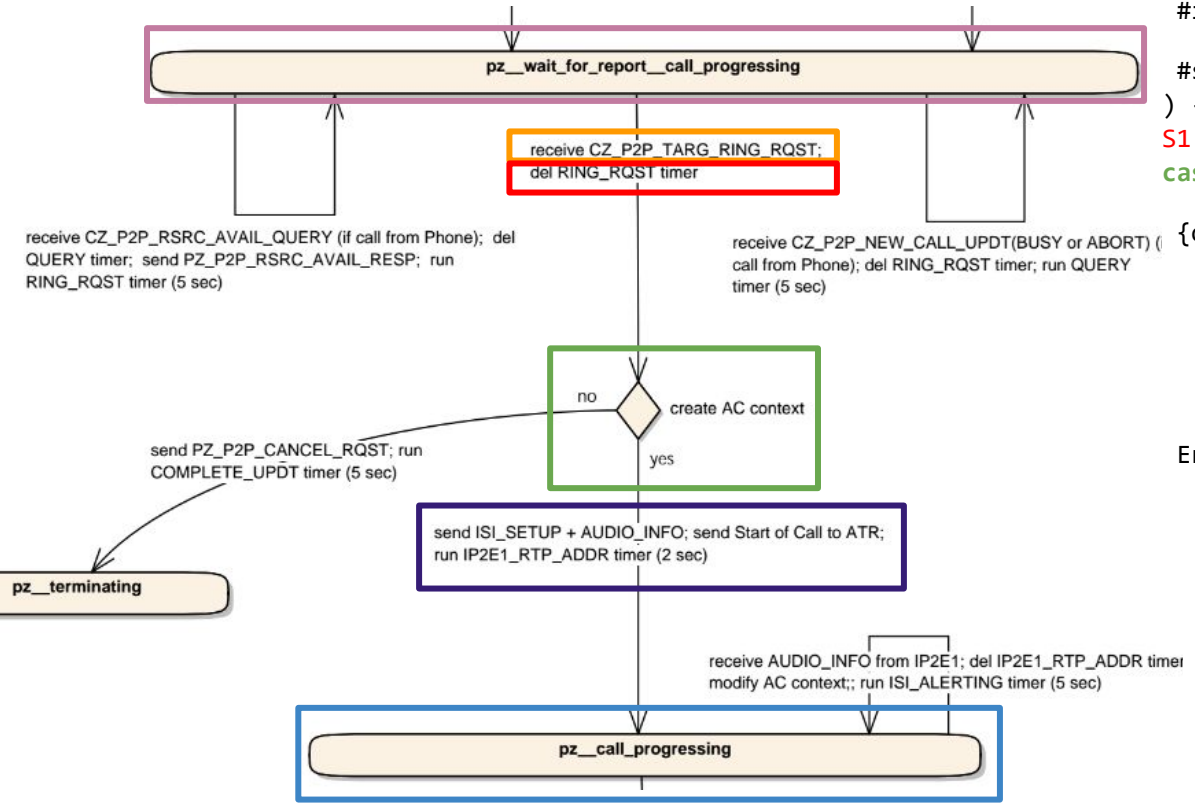



```
call_initiated(#cc_release_request{}=Msg,
   #state{call_reference=CallReference,
                          call_type=cc}=State) ->
    isi_error:log(?MODULE, info,
     "State call_initiated
   received:~n~p~n",[Msg]),
    NewState = stop_timer(t303,State),
    outgoing_message(
   #qsig_release{call_reference=CallReference},
                  State),
    NextState = start_timer(t308,NewState),
    {next_state, release_request, NextState};
```

UML

(introduced 1994)


Behavioral State
Simple State
Simple state
Simple state name
internal activities
Composite State
Composite state
Composite state - decomposition hidden
Composite state
Submachine state
Composite state
[region]
[region]
Region
[region]
Composite icon
Protocol State Machine
state machine
Protocol state
Protocol state
Transition
Pseudostate
Initial pseudostate
Terminate pseudostate
Entry point pseudostate
Exit point pseudostate
Choice pseudostate
Fork / Join pseudostate - vertical
Fork / Join pseudostate - horizontal
Junction pseudostate
Shallow history
H
Deep history
H*
Final state




```
pz__wait_for_report__call_progressing(
 #info_container{dev_info =
     #cz_p2p_targ_ring_rqst_dev_info{}} = IzMsg,
 #state{} = State
) ->
S1 = del_temp_timer(State),
case ac_proxy:create_context(#context{rtp =
     #rtp{}, duplex_flag = Duplex}) of
 {ok, AcContextId, AcRtpIp, AcPort} ->
   IsiMsg = construct_isi_setup(S1),
   S3 = run_temp_timer(?IP2E1_RTP_ADDR_TMO,
     ip2e1_rtp_addr_tmo, S2),
   {next_state, pz__call_progressing,
     S3#state{ac_context_id = AcContextId}}
   end;
 Error ->
   ?ERR(Cti, "~p: ac_proxy:create_context() error:
     ~p", [StateName, Error]),
   S2 = cancel_call(cancel_by_isi_gw,
     ?R_C_RSN_TETRA_CAUSE_NOT_DEF, S1, StateName),
   {next_state, pz__terminating, S2}
```

ErlEsy - demo

Our tool

- Auto-generates gen_fsm to .dot and .json
- Requires include files for code parser
- Supports
  - Simple state transitions
  - Transitions within conditionals
  - Guards
  - All State events (simplified and expanded)
- https://github.com/haljin/erlesy

Jobs Jobs Jobs

JOB OPENING – Motorola Solutions

Software Developers

Motorola Solutions is a leading supplier of smart work group communication solutions for enterprises and governments around the world. We focus exclusively on professional markets such as public safety, utilities, energy, transportation, manufacturing, and other commercial and industrial customers.

The Job

You will join a team of highly skilled software engineers that drive the core functionality development and make the network infrastructure of Motorola Solutions' mission critical communication systems. Based in the Copenhagen office, you will open mindedly share ideas and proactively collaborate with customers and colleagues locally as well as across the globe to create strong technical and customer-oriented solutions.

The Person

You have a strong desire to develop market leading products to the most demanding users in the world as well as the potential and ambition to be one of the best software engineers. It will take energy and drive to be successful and we are looking for someone who is ready to challenge our existing ways of working.

In addition, these are the characteristics we are looking for:

- Recently graduated or with a few years of experience, possibly from a software based startup company.
- Expertise in real time Linux server solutions.
- Good knowledge of C/C++ and/or functional programming (**Erlang** or similar).
- Strong ability and enthusiasm to learn new technologies in a short time