

1/45

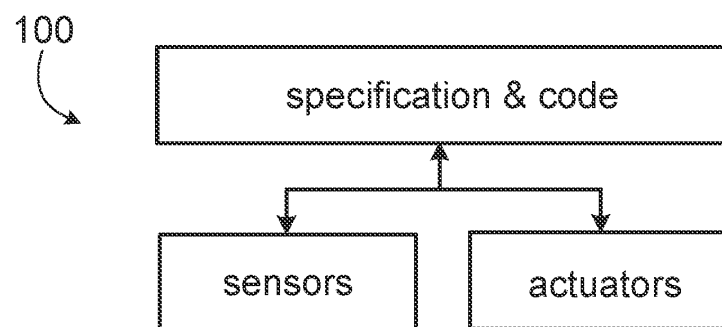


Fig. 1

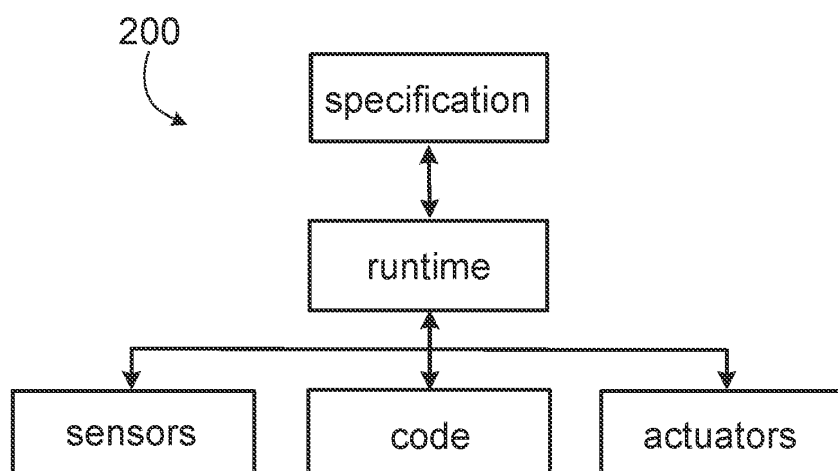


Fig. 2

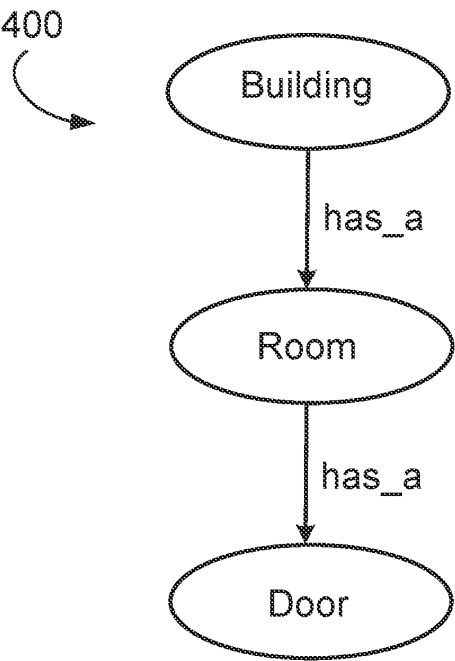


Fig. 4

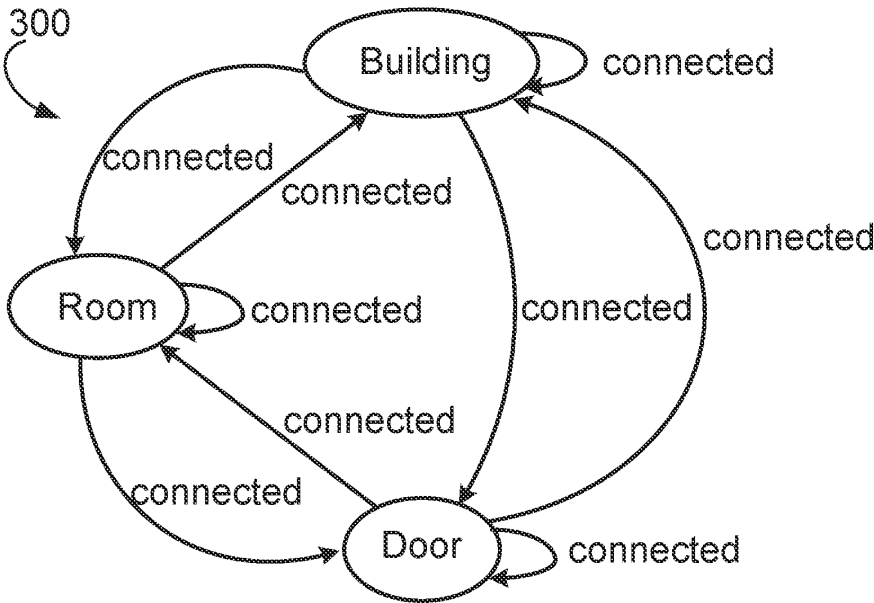
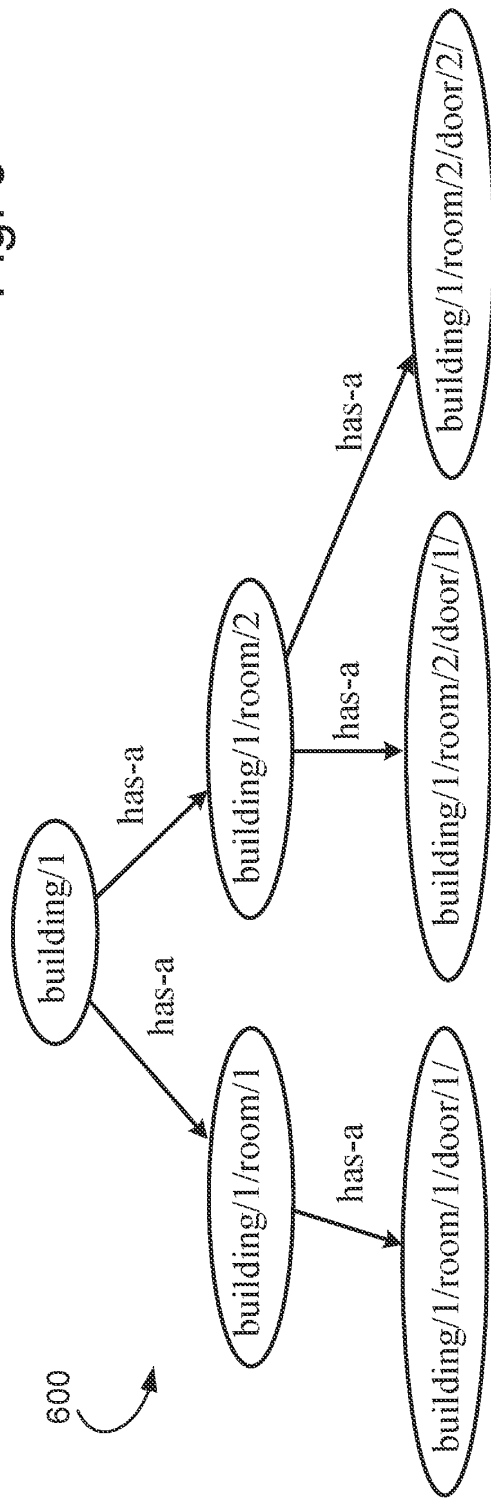
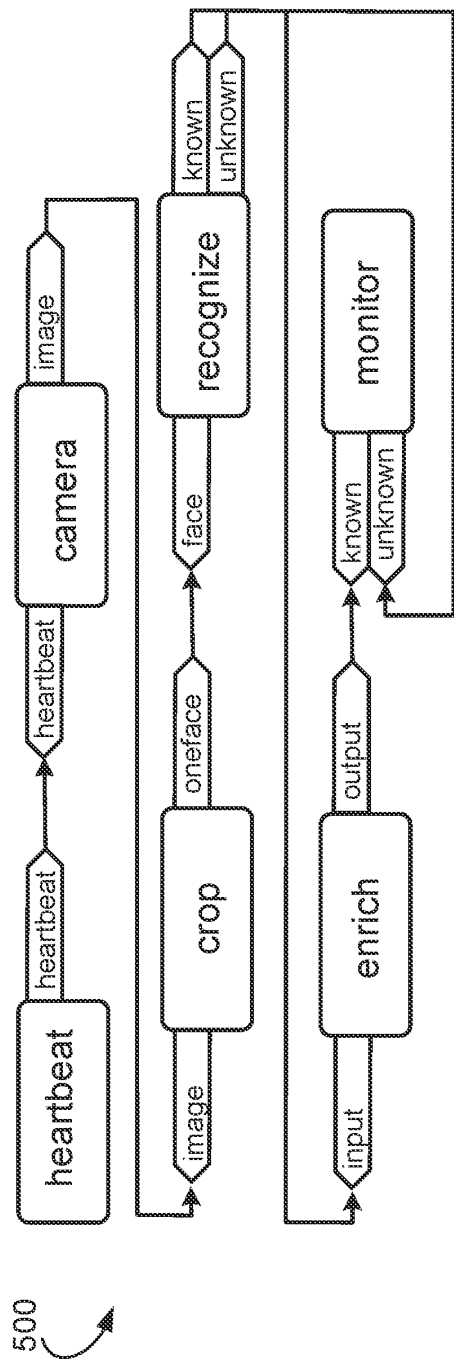


Fig. 3



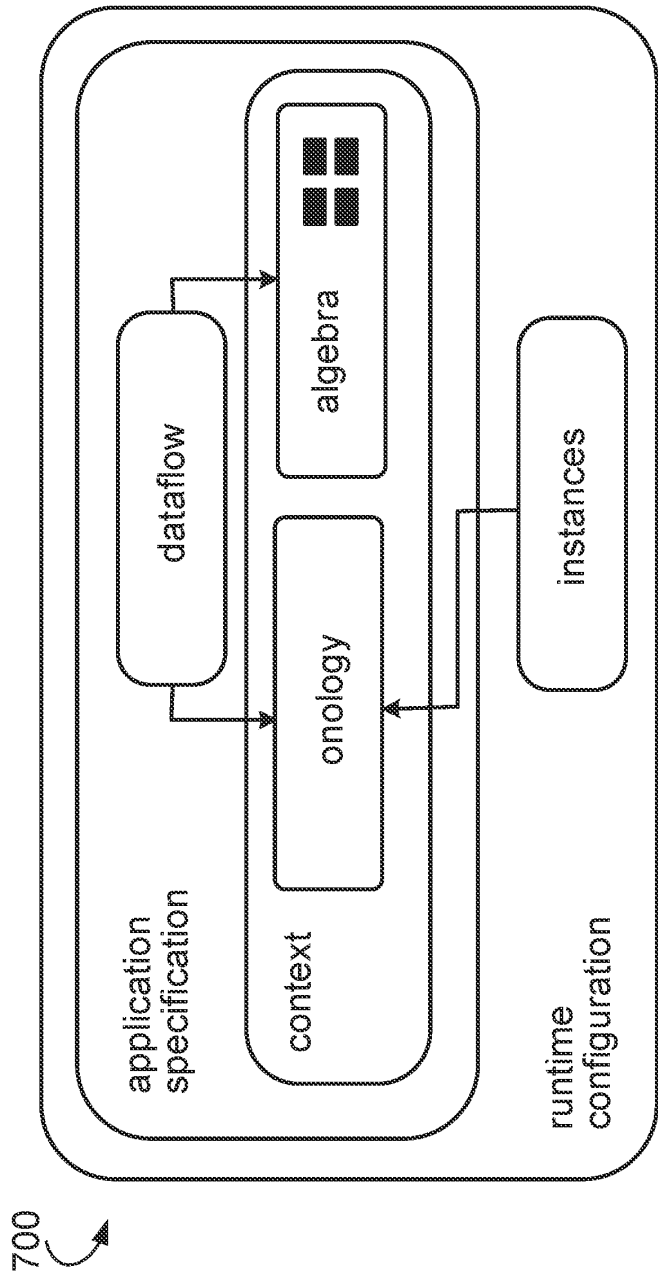


Fig. 7

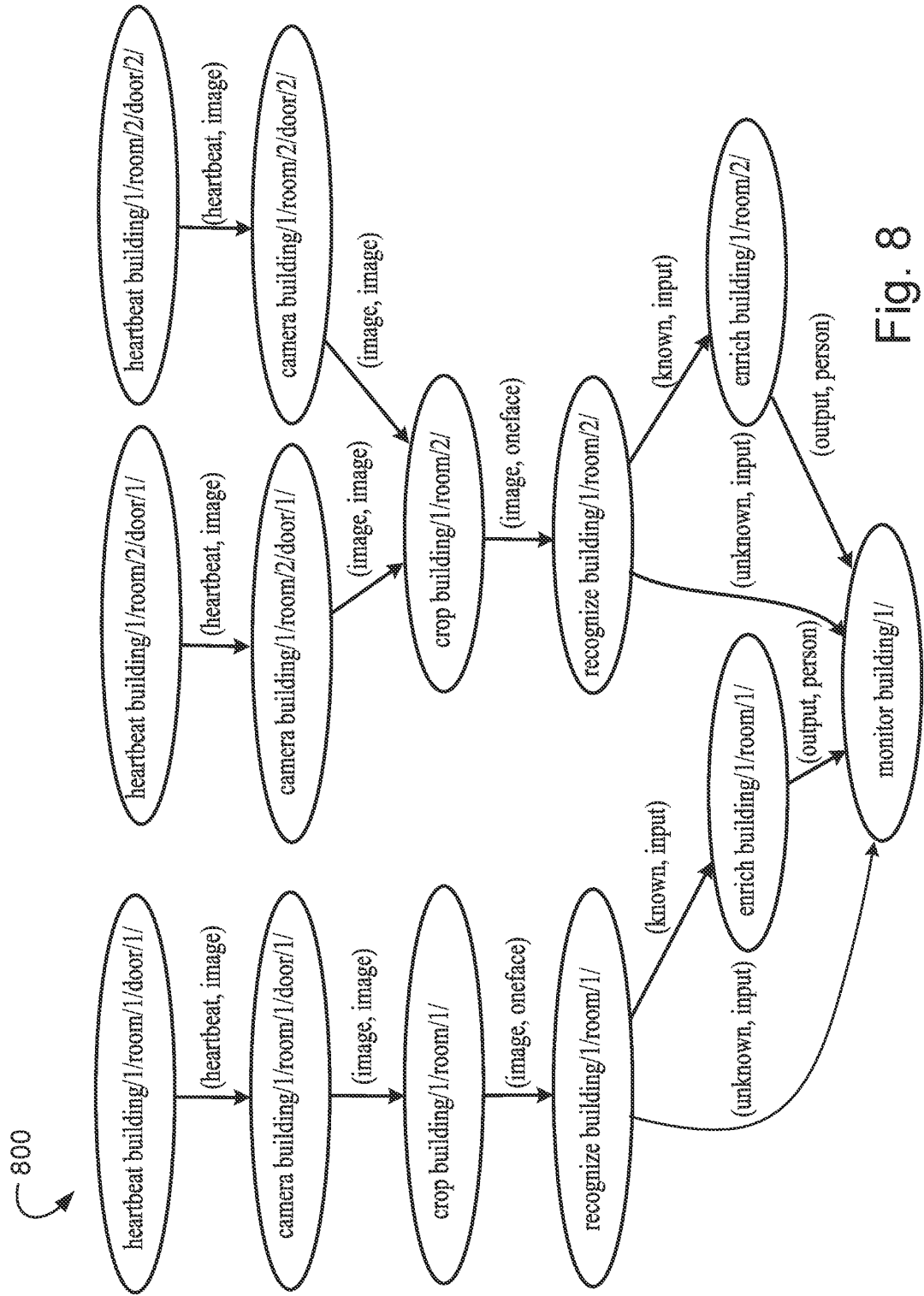


Fig. 8

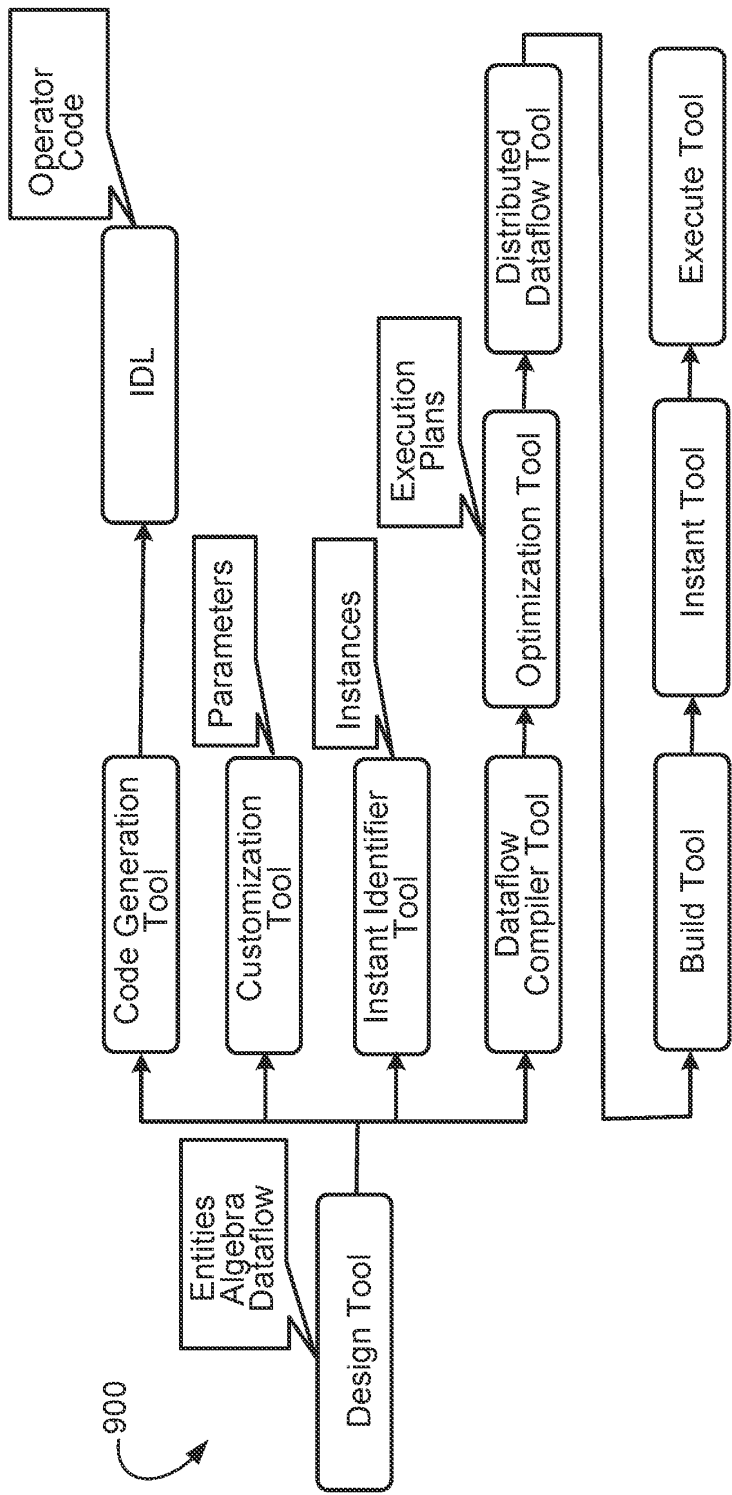


Fig. 9

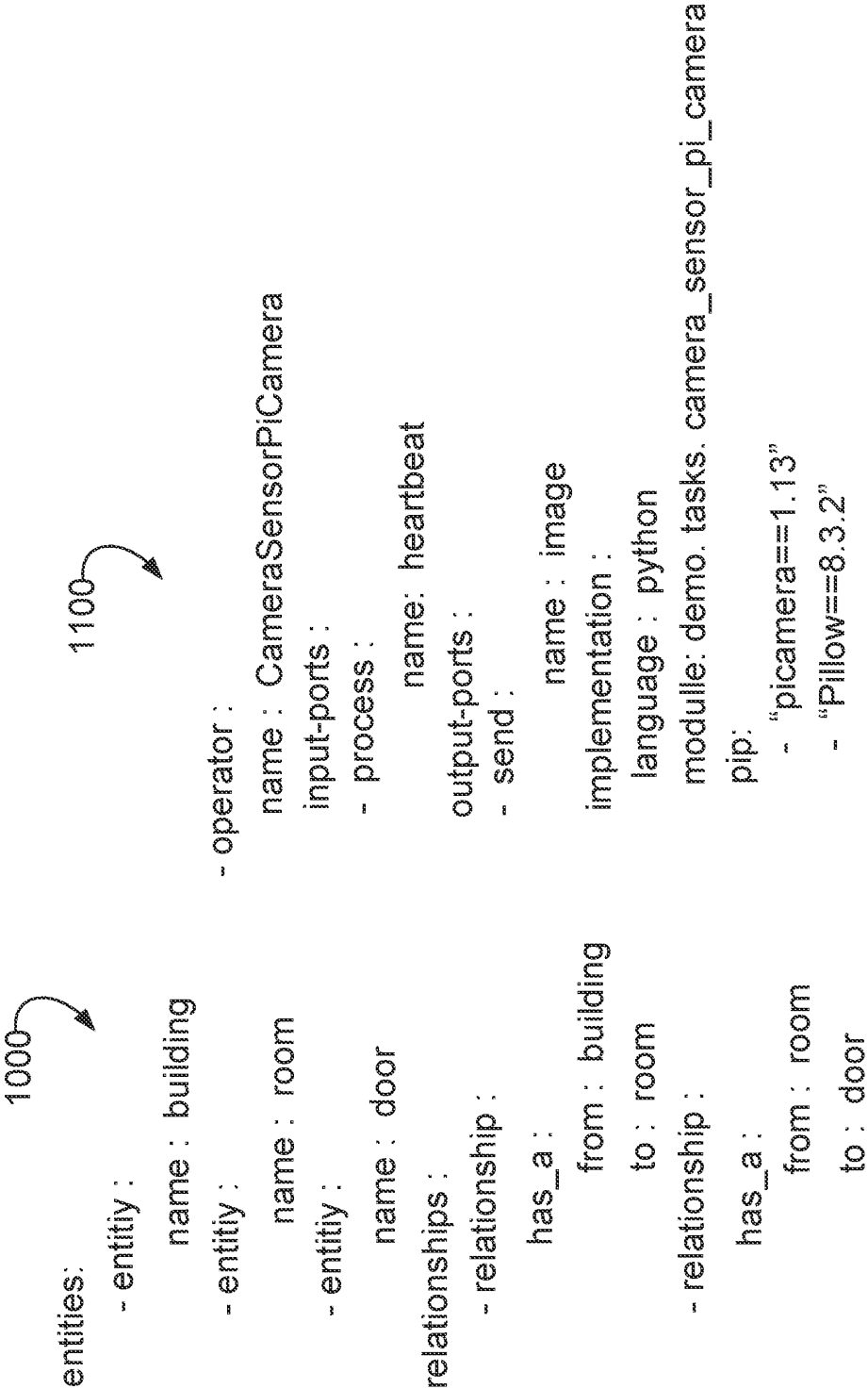


Fig. 10

Fig. 11

8/45

1200



- tasks:
 - task: { name: camera, operator: CameraSensorPiCamera, scope: door }
 - task: { name: crop, operator: Crop, scope: door }
 - task: { name: recognize, operator: Recognize, scope: room }
 - task: { name: heartbeat, operator: Heartbeat, scope: door }
 - task: { name: enrich, operator: Enrich, scope: room }
 - task: { name: monitor, operator: Monitor, scope: building }
- connections:
 - connection:
 - relationship: related
 - from: { task: heartbeat, output_port: heartbeat }
 - to: { task: camera, input_port: heartbeat }
 - connection:
 - relationship: related
 - from: { task: camera, output_port: image }
 - to: { task: crop, input_port: image }
 - connection:
 - relationship: related
 - from: { task: crop, output_port: oneface }
 - to: { task: recognize, input_port: image }
 - connection:
 - relationship: related
 - from: { task: recognize, output_port: known }
 - to: { task: enrich, input_port: input }
 - connection:
 - relationship: related
 - from: { task: recognize, output_port: unknown }
 - to: { task: monitor, input_port: image }

Fig. 12

9/45

- instances:
 - instance: { id: building/1/, entity: building }
 - instance: { id: building/1/room/1/, entity: room }
 - instance: { id: building/1/room/2/, entity: room }
 - instance: { id: building/1/room/1/door/1/, entity: door }
 - instance: { id: building/1/room/2/door/1/, entity: door }
 - instance: { id: building/1/room/2/door/2/, entity: door }
- relationships:
 - relationship:
 - name: has-a
 - from: building/1
 - to: building/1/room/1/
 - relationship:
 - name: has-a
 - from: building/1
 - to: building/1/room/2/
 - relationship:
 - name: has-a
 - from: building/1/room/1/
 - to: building/1/room/1/door/1/
 - relationship:
 - name: has-a
 - from: building/1/room/2/
 - to: building/1/room/2/door/1/
 - relationship:
 - name: has-a
 - from: building/1/room/2/
 - to: building/1/room/2/door/2/

1300


Fig. 13

10/45

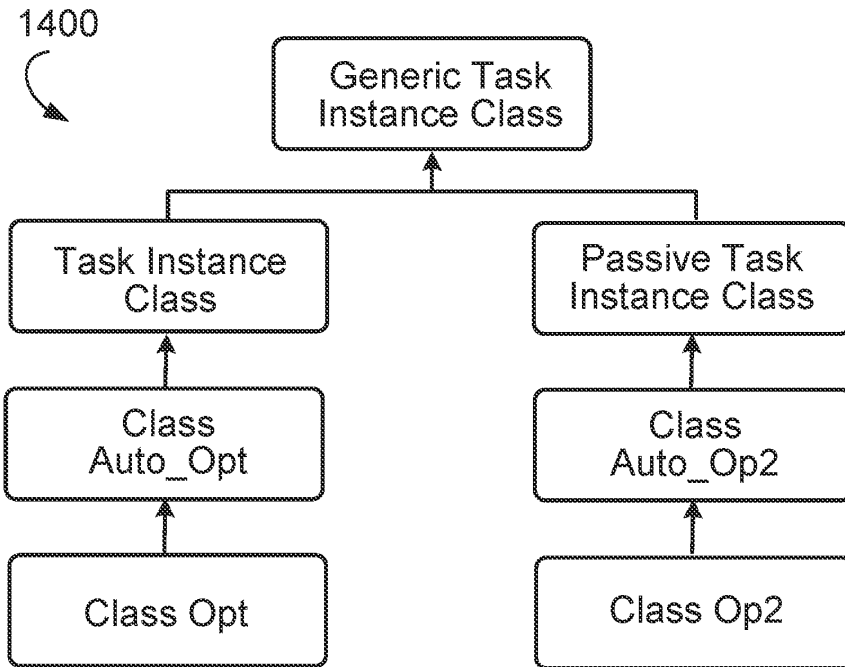


Fig. 14

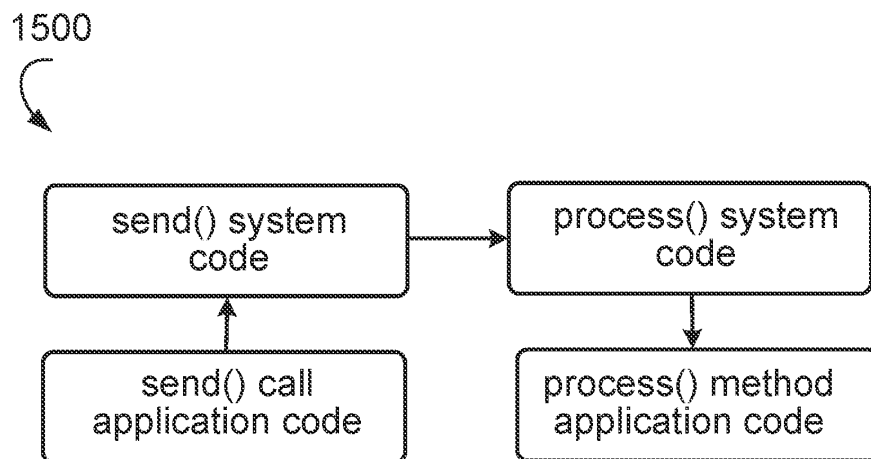


Fig. 15

11/45

1600
output – ports :
- send :
 name : heartbeat
implementation :
 language : python
 strategy : active
 module : autocoder . library . operator . heartbeat

1602
def _run_once_magbe (self) :
 self . event . wait (self . delay)
 self . send _ heartbeat ([self . event_factory ()])

Fig. 16

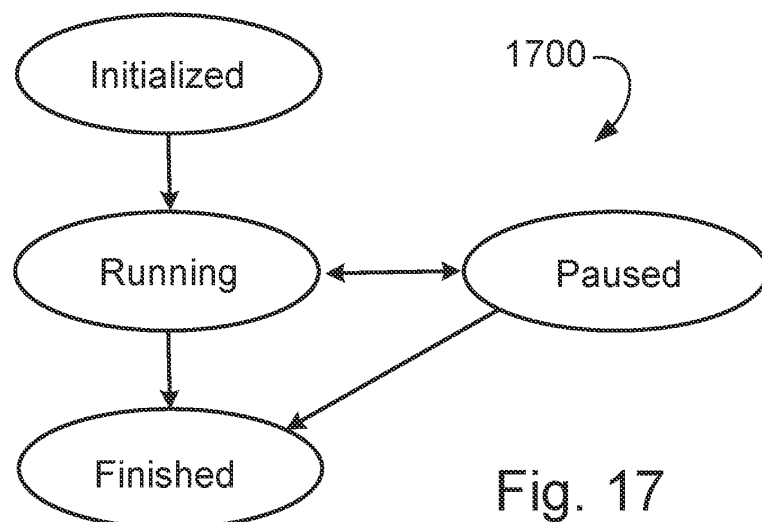


Fig. 17

12/45

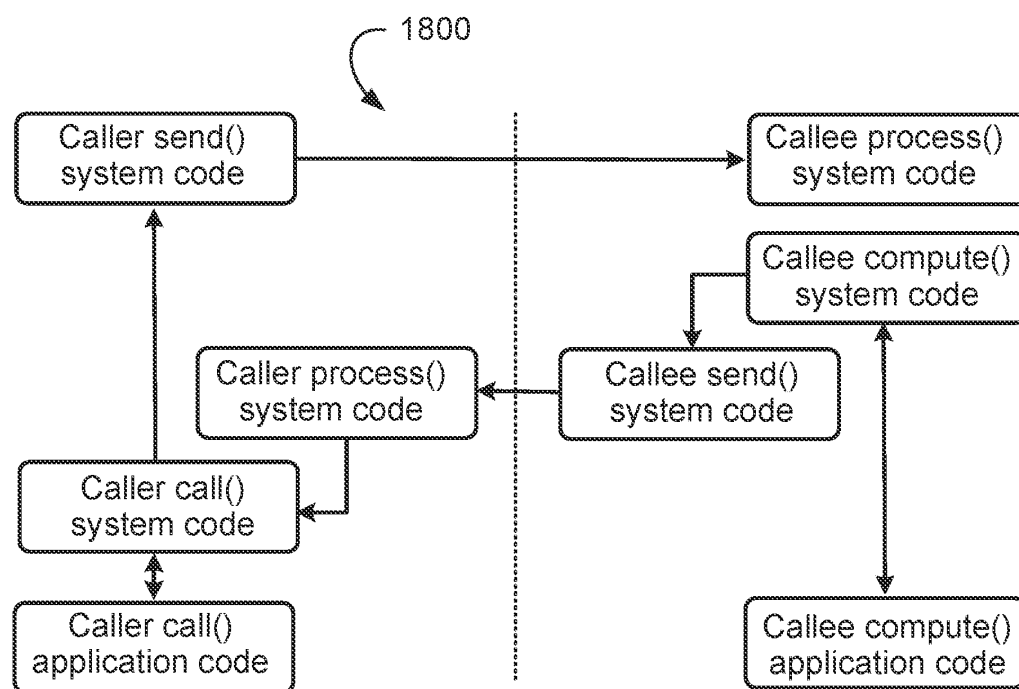


Fig. 18

13/45

```

class Caller(ABC):
    def call(self, port_name, input):
        call_id = self.mk_call_id(port_name)
        e = (call_id, input, self.instance.instance_id)
        self.replies[call_id] = queue.Queue()
        self.send('.'.join([port_name, 'request', 'start']), [Event(e)])
        ans = self.replies[call_id].get()
        del self.replies[call_id]
        return ans

    def general_process_reply_end(self, port_name, event):
        (call_id, result, caller_id) = event.get_payload()
        q = self.replies.get((port_name, call_id))
        if q not null:
            q.put(result)

```

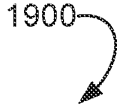


Fig. 19

```

class Callee(ABC):
    def general_process_request_start(self, port_name, event):
        (call_id, input, caller_id) = event.get_payload()
        fun = getattr(self, 'compute_' + port_name)
        result=fun(portname, input)
        e = (call_id, result, caller_id)
        self.send('.'.join([port_name, 'reply', 'end']),
                  Event(e),
                  caller_id)

```

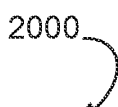
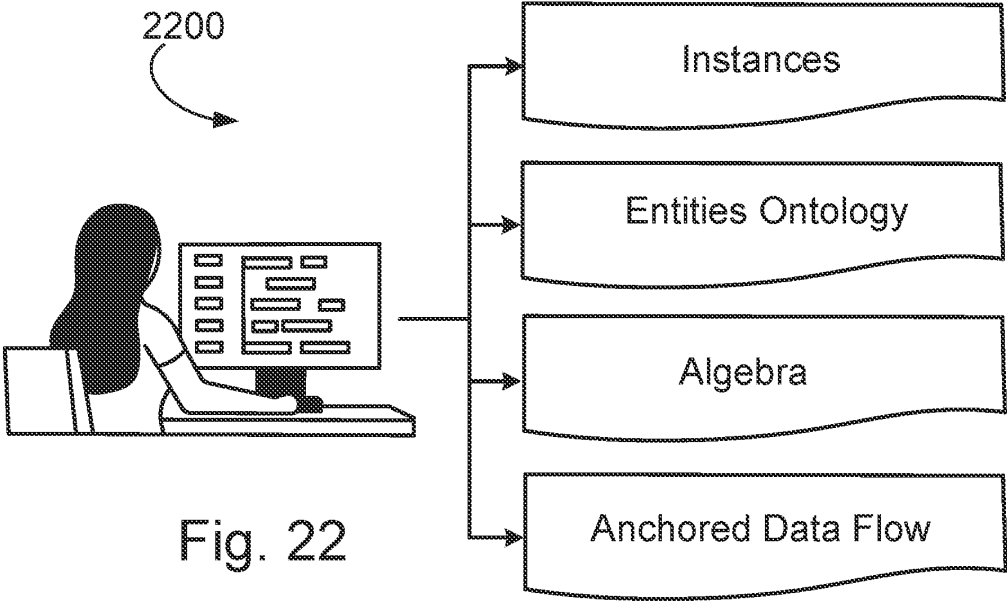
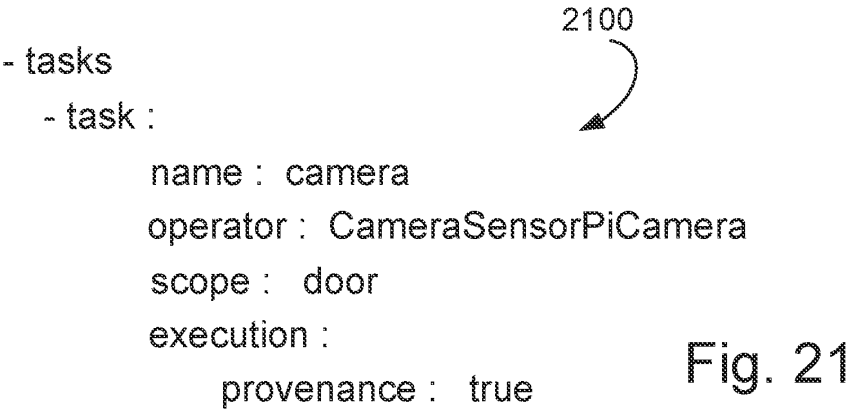


Fig. 20



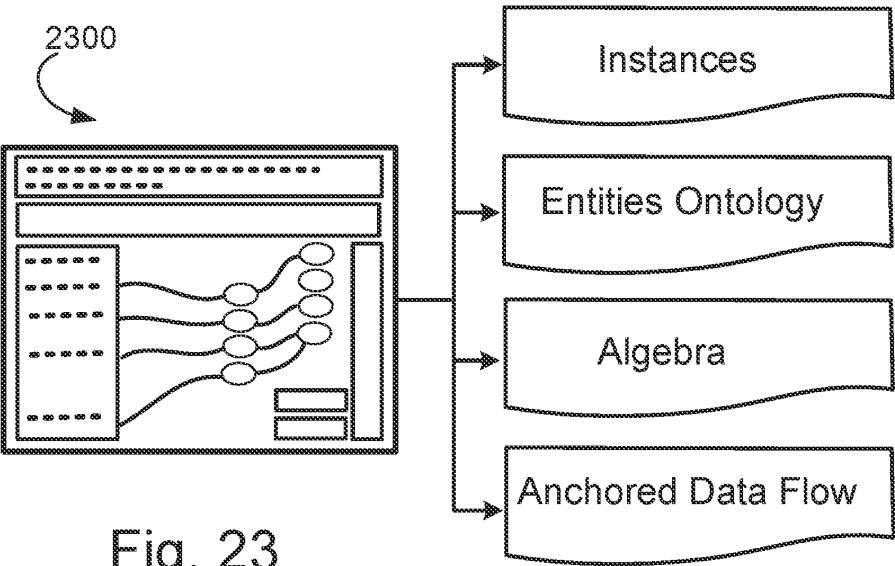


Fig. 23

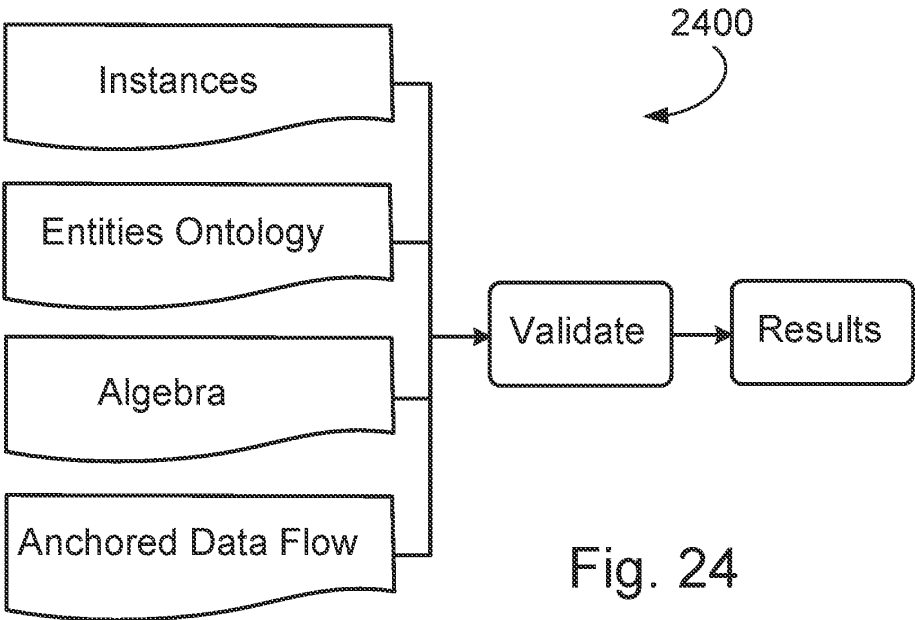


Fig. 24

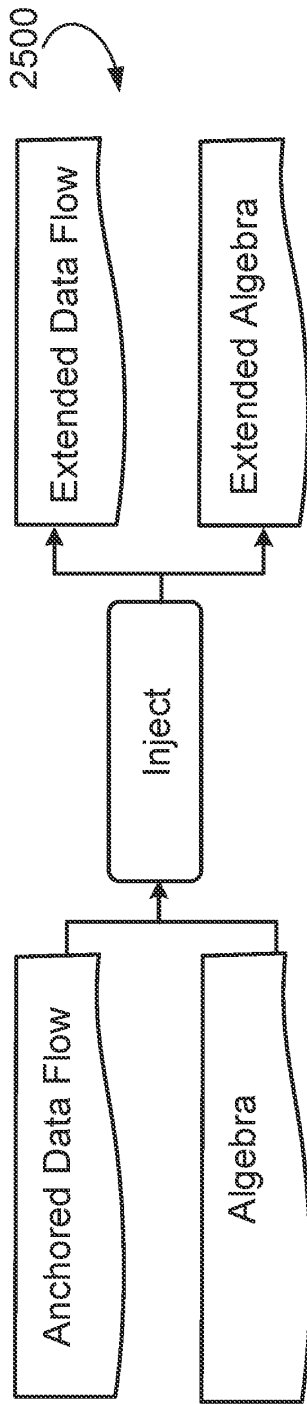


Fig. 25

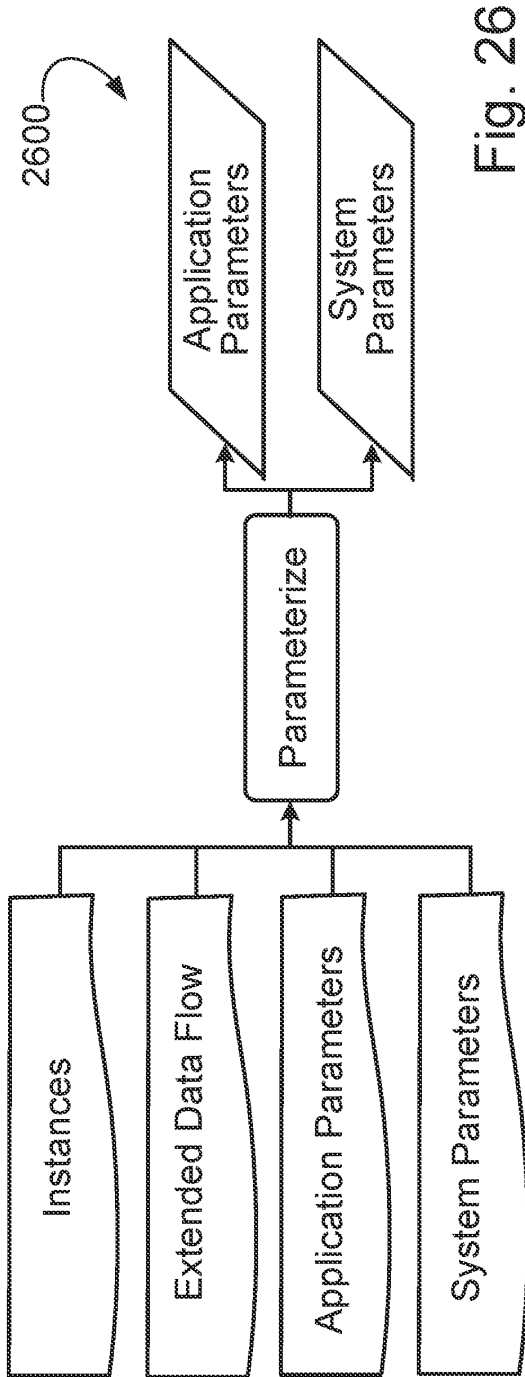


Fig. 26

- cluster:

name: cluster_one

tasks: [heartbeat, camera]

- cluster:

name: cluster_two

tasks: [crop, recognize, enrich, monitor]

2700

Fig. 27

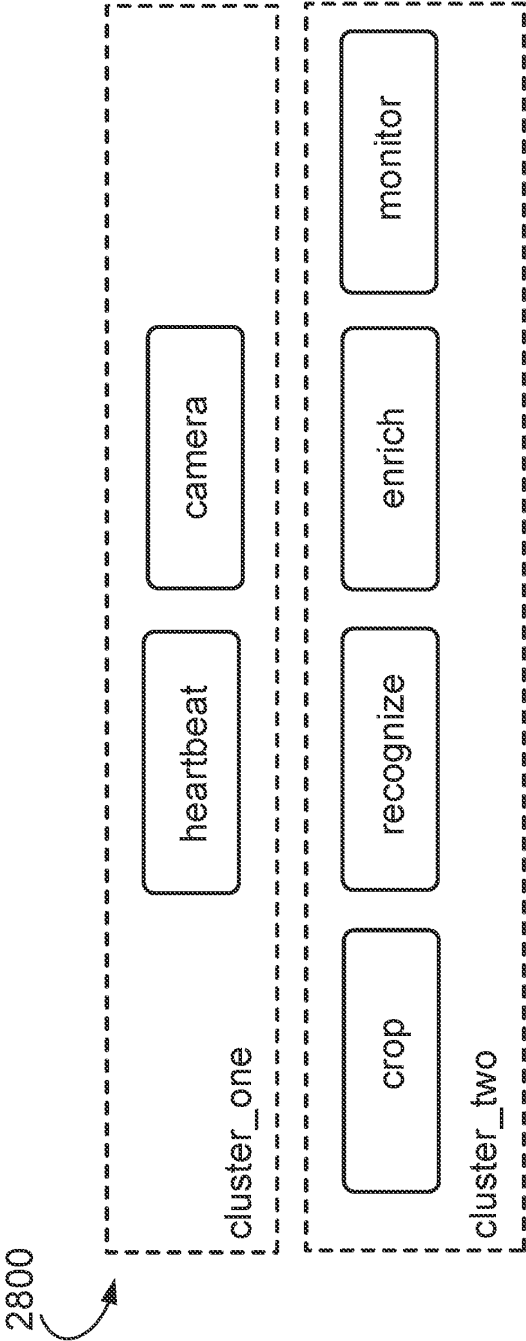


Fig. 28

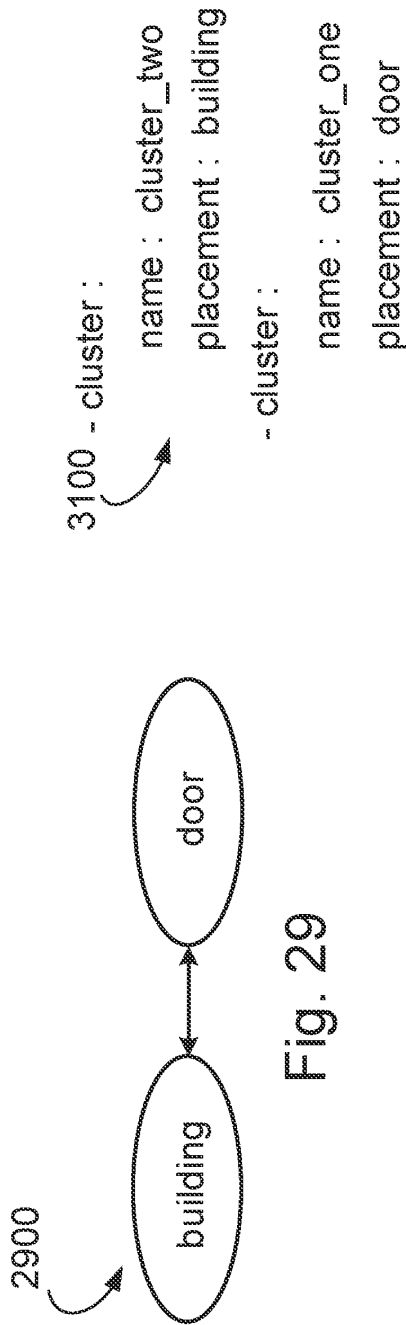


Fig. 31

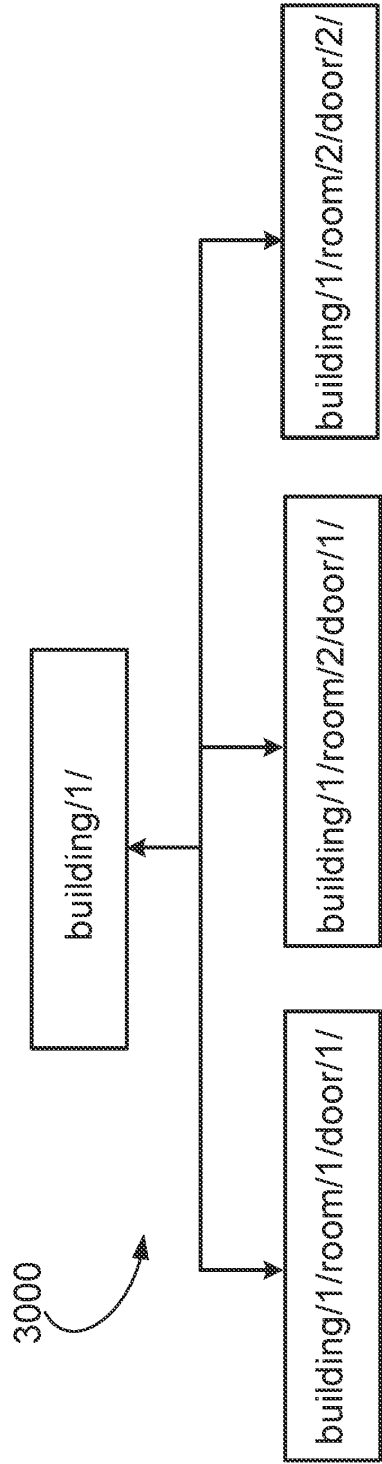


Fig. 30

19/45

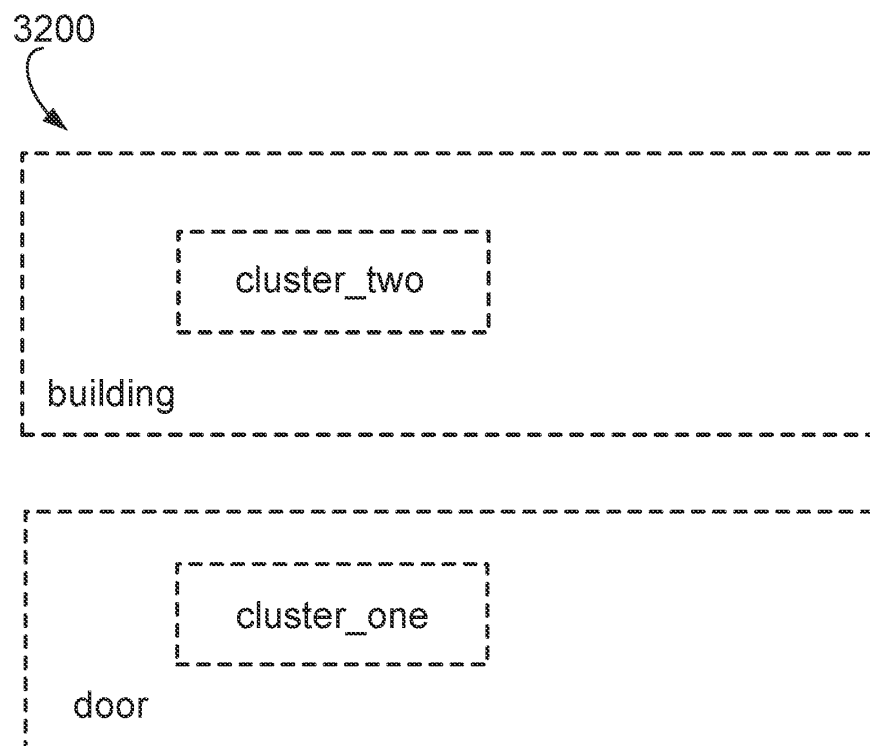


Fig. 32

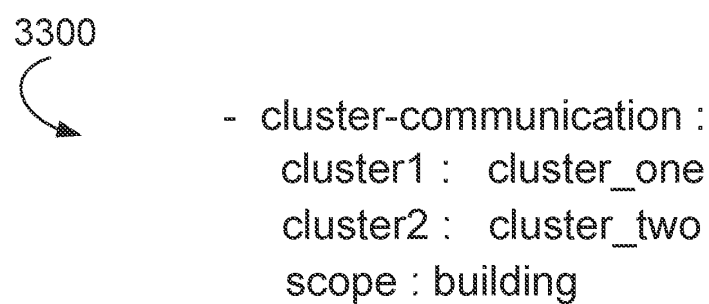
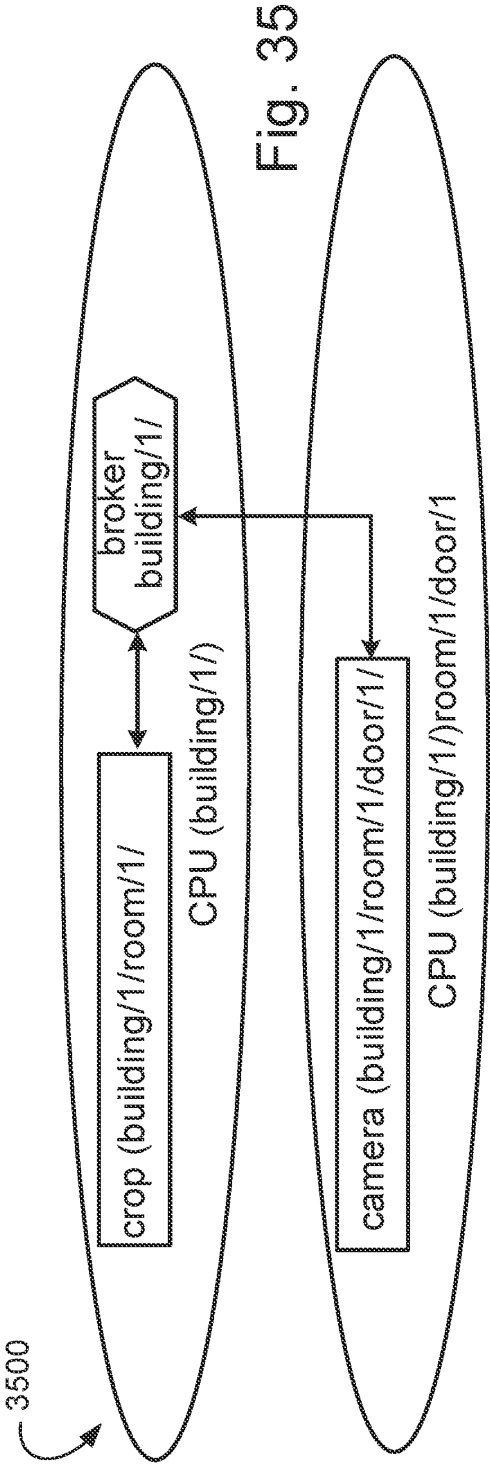
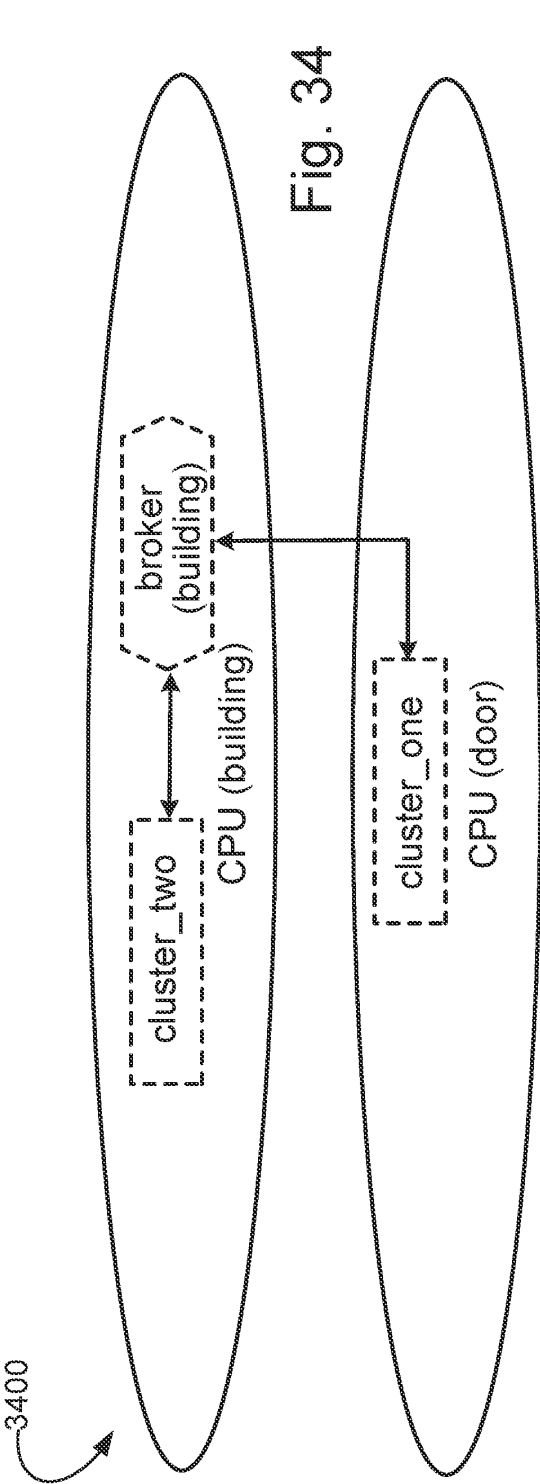


Fig. 33



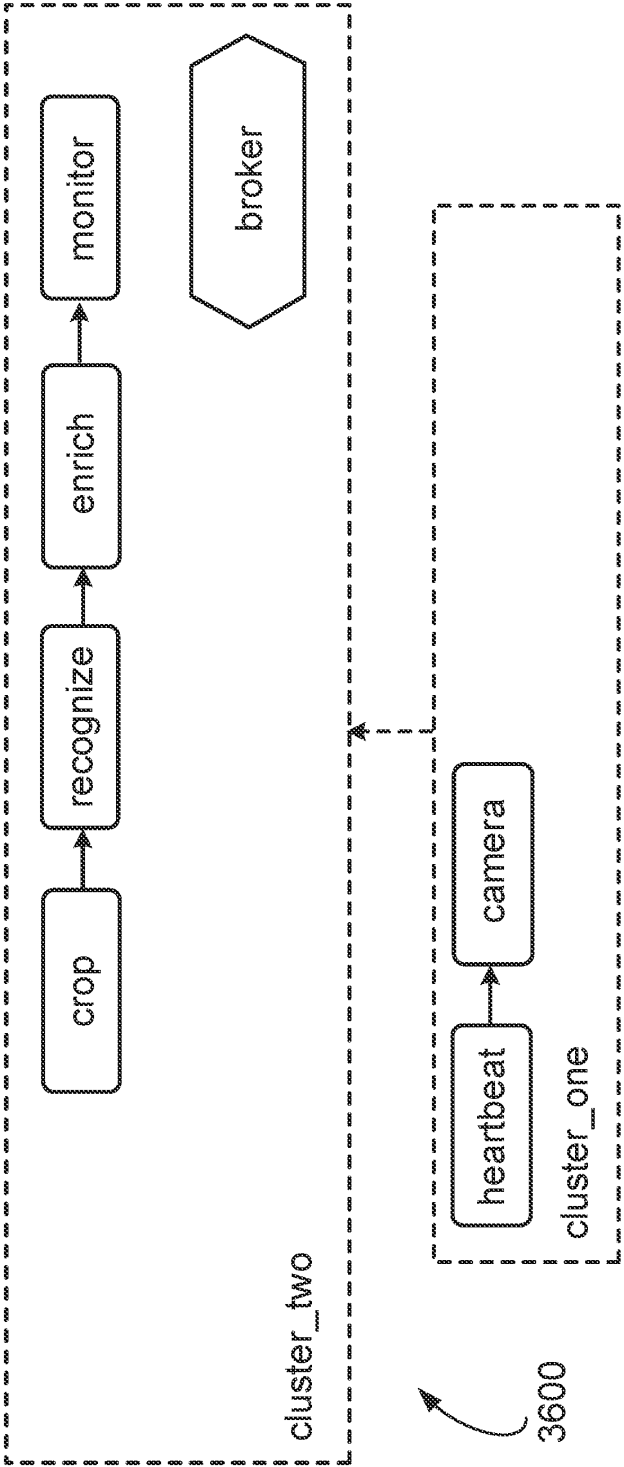


Fig. 36

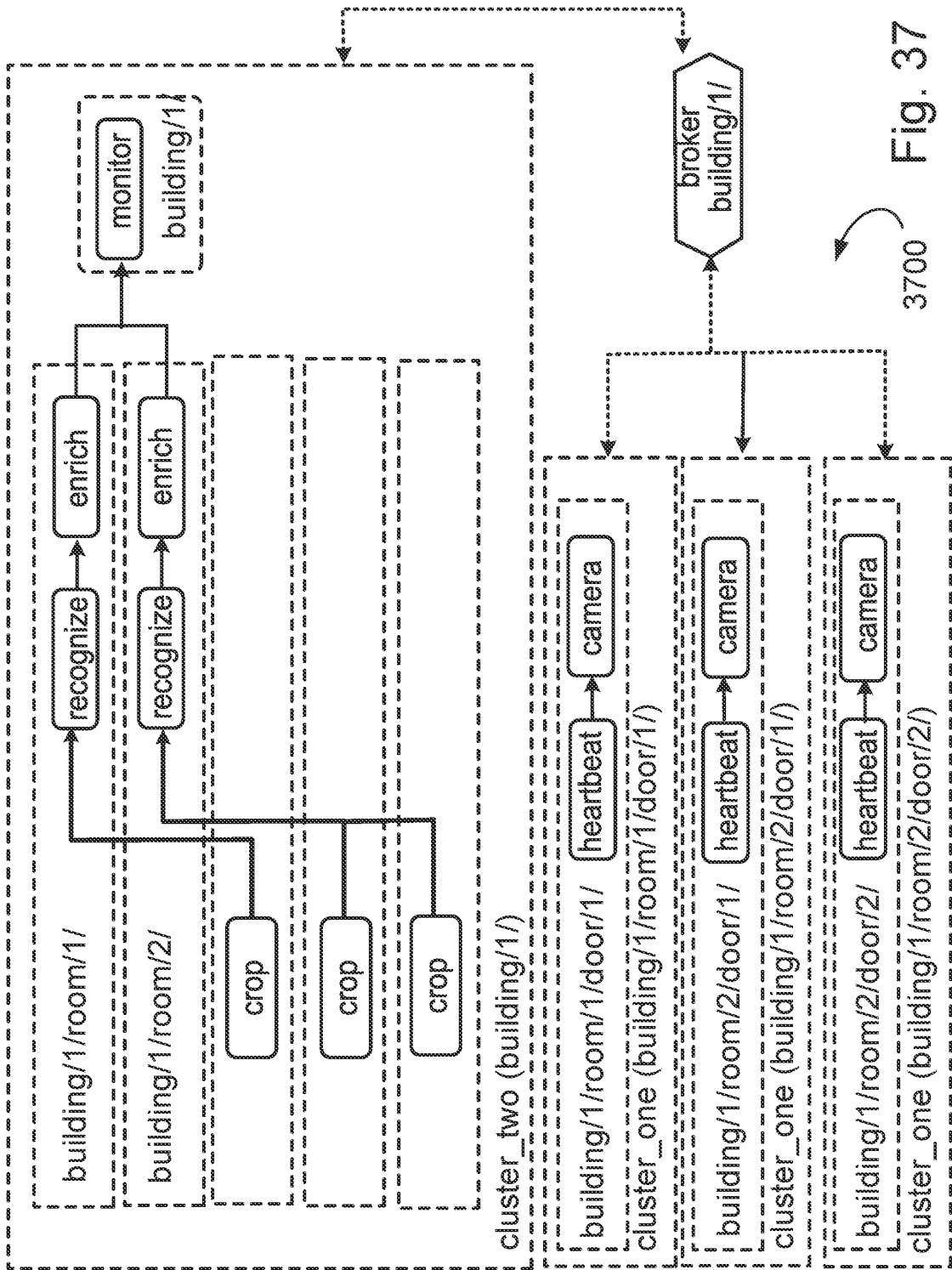


Fig. 37

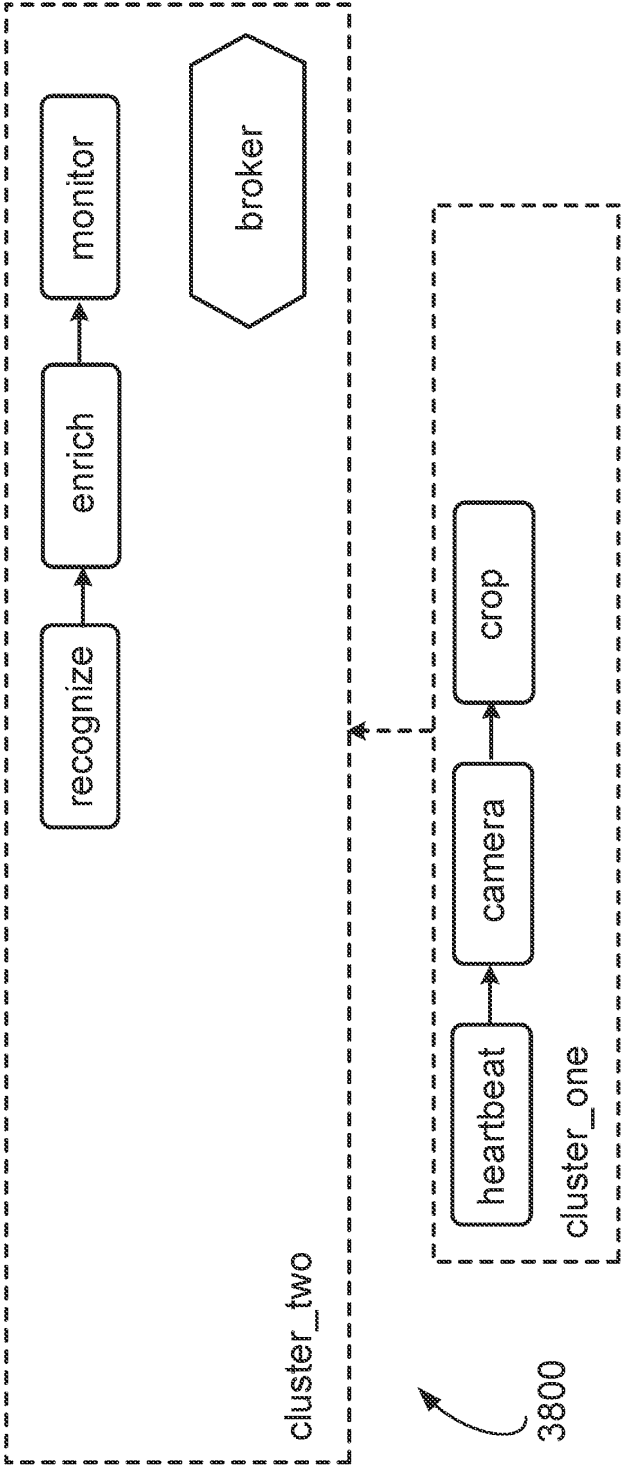


Fig. 38

24/45

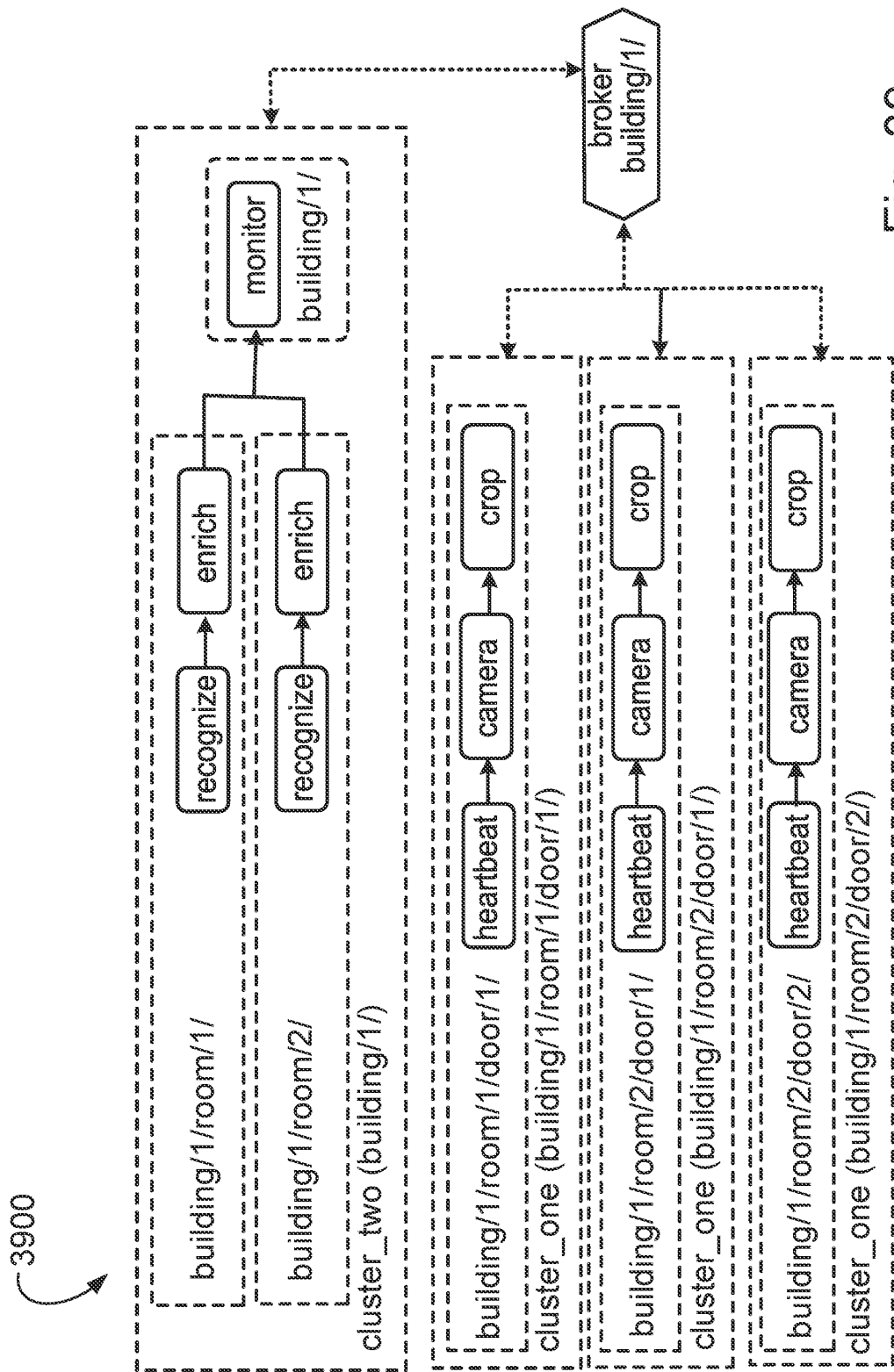


Fig. 39

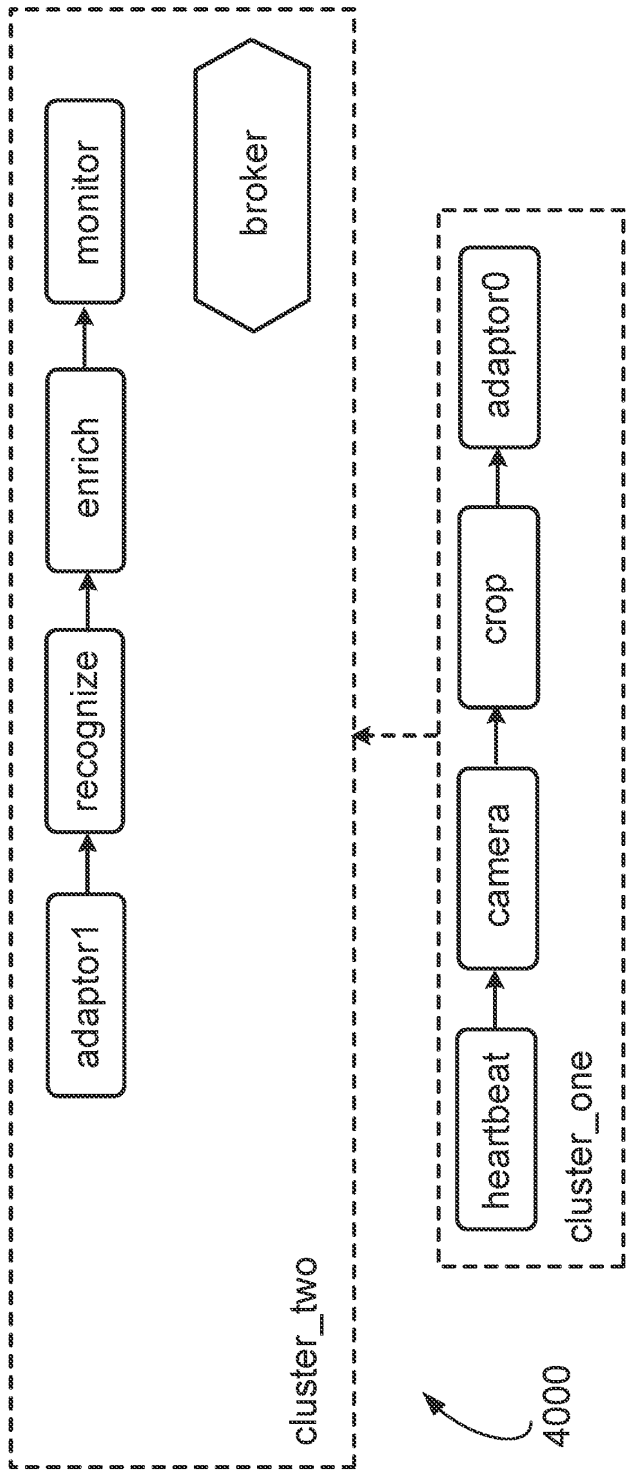
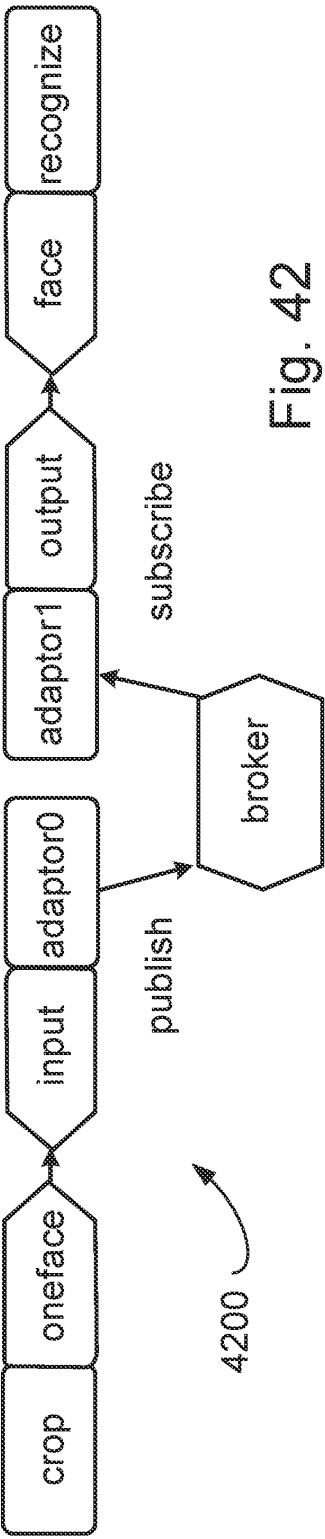
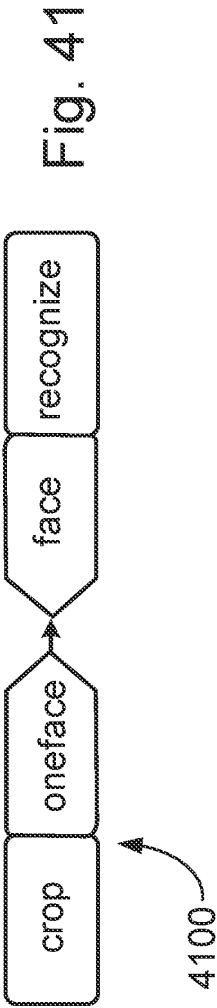


Fig. 40



27/45

4300



```
- operator :  
  name : CameraSensorPiCamera  
  input - ports :  
  - process :  
      name : heartbeat  
  output - ports :  
  - send :  
      name : image  
  implementation :  
    language : python  
    module : demo . task . camera_sensor-pi-camera  
    schema :  
      input - ports :  
        name : heartbeat  
        type : int  
      output - ports :  
        name : image  
        type : Image  
  pip :  
    - " picamera==1.13 "  
    - " Pillow==8.3.2 "
```

Fig. 43

28/45

4400

```

- name: initial_state
  value: state_apple

- name: transition_matrix
  value: { state_apple: { 0: state_apple, 1: state_orange, 2: state_orange },
          state_orange: { 0: state_apple, 1: state_apple, 2: state_apple }}

```

Fig. 44

4500

```

def classify(self, event):
    n = event.get_payload()
    return (event.get_payload()) % 3

def state_apple(self, event, old_state, new_input, new_state):
    self.write_apple([Event(str(event.get_payload()))])

def state_orange(self, event, old_state, new_input, new_state):
    self.write_apple([Event(str(event.get_payload()))])

```

Fig. 45

29/45

```

def process_input(self, event):
    self.old_state = self.state
    self.new_input = self.classify(event)
    self.new_state = self.transition_matrix[self.state][self.new_input]

    fun = getattr(self, self.new_state)
    fun(event, self.old_state, self.new_input, self.new_state) self.state =
    self.new_state

```

Fig. 46

4600

```

def call_execute(self, dbrequest):
    with self.connection.cursor() as cursor:
        cursor.execute(dbrequest.query, dbrequest.parameters)
        return cursor.fetchall()

```

Fig. 47

4700

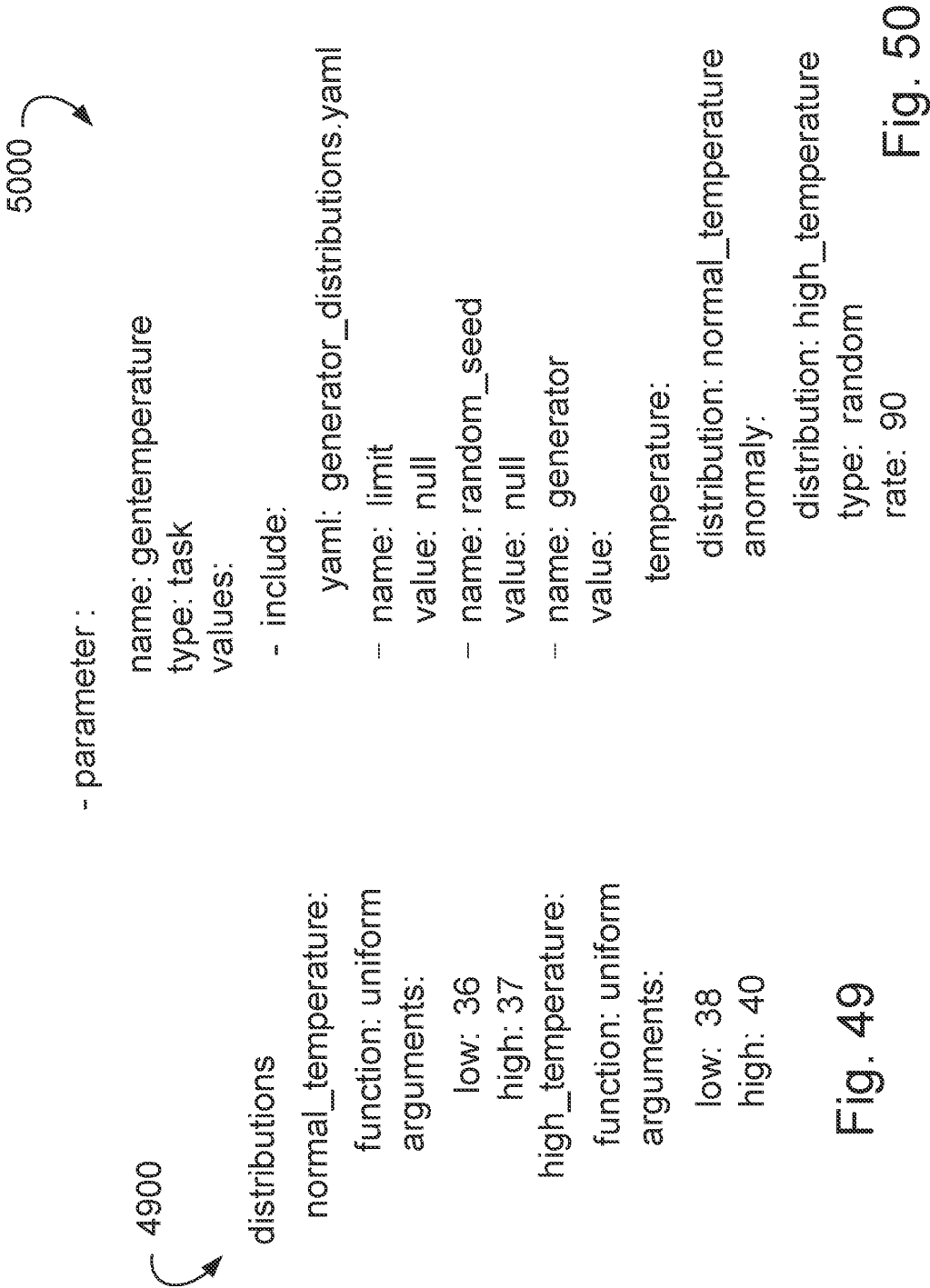
4800

```

self.write_output(self.merge(event, self.call_fetch(self.project(event))))

```

Fig. 48



31/45

5100

```

def process_input(self, event):
    for port in self.event_condition_action_output_ports:
        condition_function = getattr(self, "condition_" + port)
        if condition_function(event):
            transform_function = getattr(self, "transform_" + port)
            self.write(port, transform_function(event))

```

Fig. 51

5200

```

def query(self):
    return [ Event("room full: " + len(w) + " people") for
            tumbling window w in self.input()
            start x at s when True
            end y at e when (timestamp(y)-timestamp(x)) > self.window_seconds
            where len(w) > self.window_limit ]

```

Fig. 52

32/45

```
class CameraSensorPiCamera(AutoCameraSensorPiCamera):
```

```
    def __init__(self, *params, **kw_params):
        super().__init__(*params, **kw_params)
        self._camera = None
        self._stream = None
        self._image = None
```

5300



```
    def process_heartbeat(self, _):
        self._stream = BytesIO()
        self._camera.capture(self._stream, format='jpeg')
        self._stream.seek(0)
        self._image = PillowImage.open(self._stream)
        self.write_image([Event(Image(image=self._image))])
```

```
    def init_resources(self):
        self._camera = PiCamera()
        self._camera.resolution = (self.camera_width, self.camera_height)
        self._camera.start_preview()
        sleep(2) # camera warm up
```

```
    def stop_resources(self):
        self._camera.close()
```

Fig. 53

5500

- dashboard :
 name : dashboard
 description : test dashboard
 scope : building
 placement : building
 port : 8000
 threaded_parallelism : 10
data :
- connection :
 from : { task : heartbeatcamera, output_port : heartbeat }
- compute :
 compute : { task : camera, port : camera_status }
- connection :
 from : { task : crop, output_port : oneface }
- connection :
 from : { task : enrichment, output_port : output }

5400

- log :
 name : test
 type : input_port
compute :
 task : check temperature
 input_port : input

Fig. 54

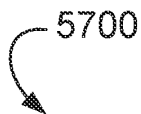
Fig. 55

34/45

5600

- merge – clusters :
- clusters : [cluster _ one , cluster _ two]
- result – cluster : cluster - three

Fig. 56

5700

- split – cluster :
- input – cluster : cluster _ two
- output
 - cluster :
 - name : cluster _ three
 - tasks : [crop , recognize]
 - cluster :
 - name : cluster - four
 - task : [enrich , monitor]

Fig. 57

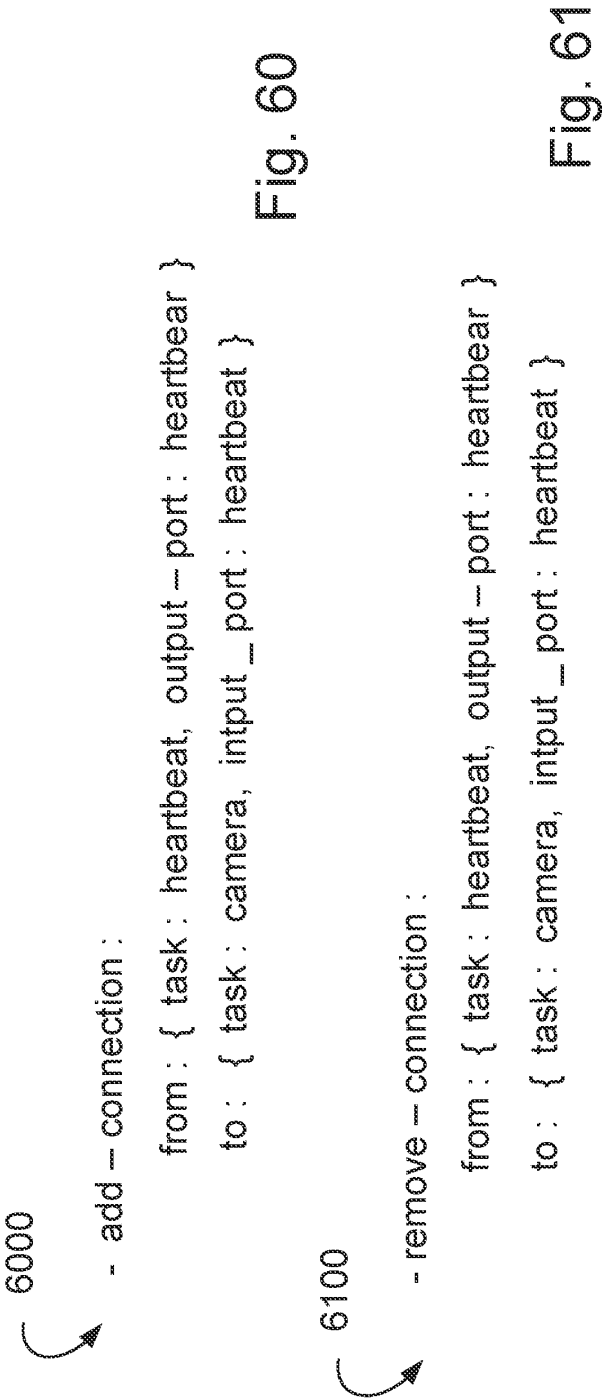
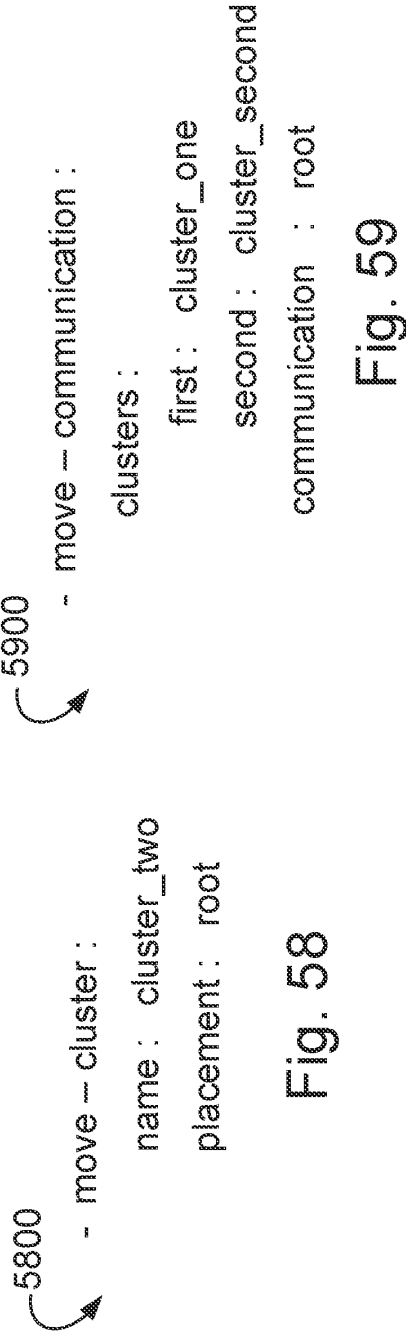




Fig. 62

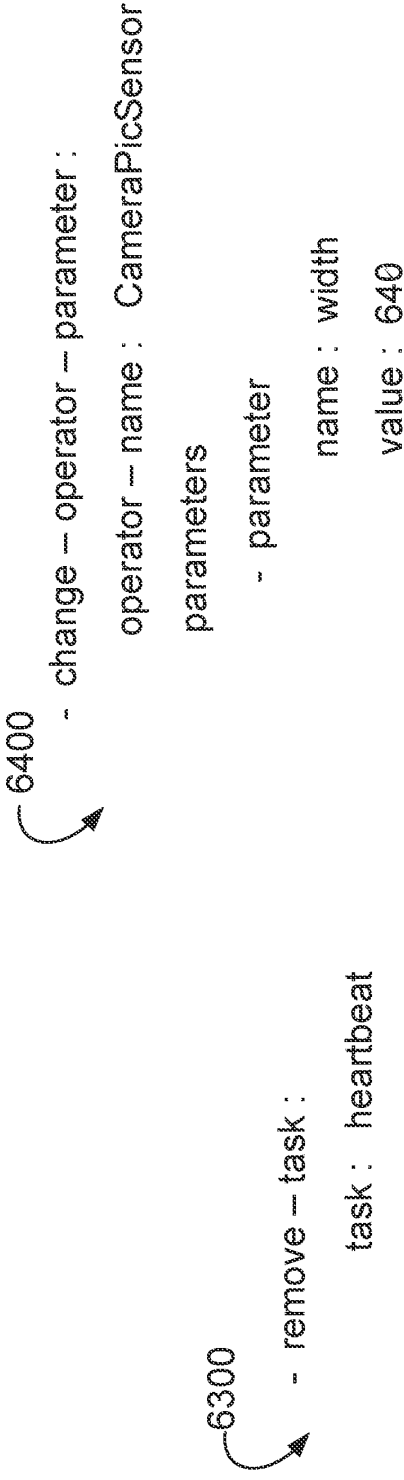
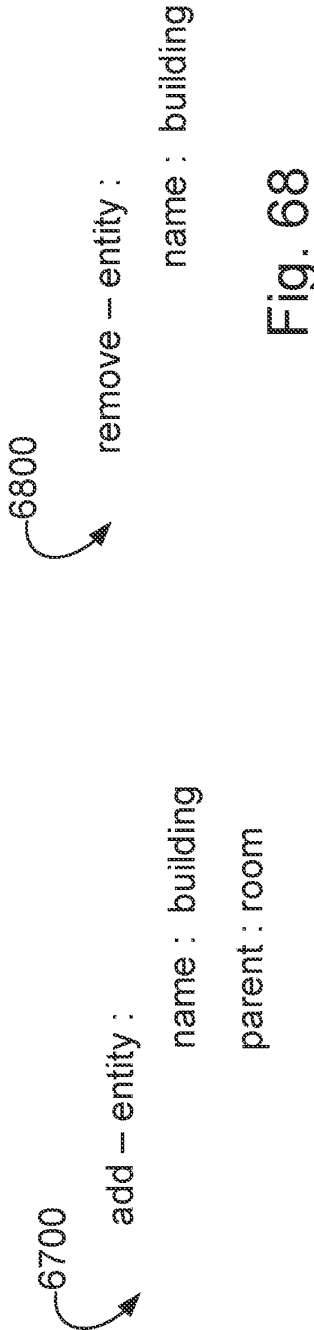
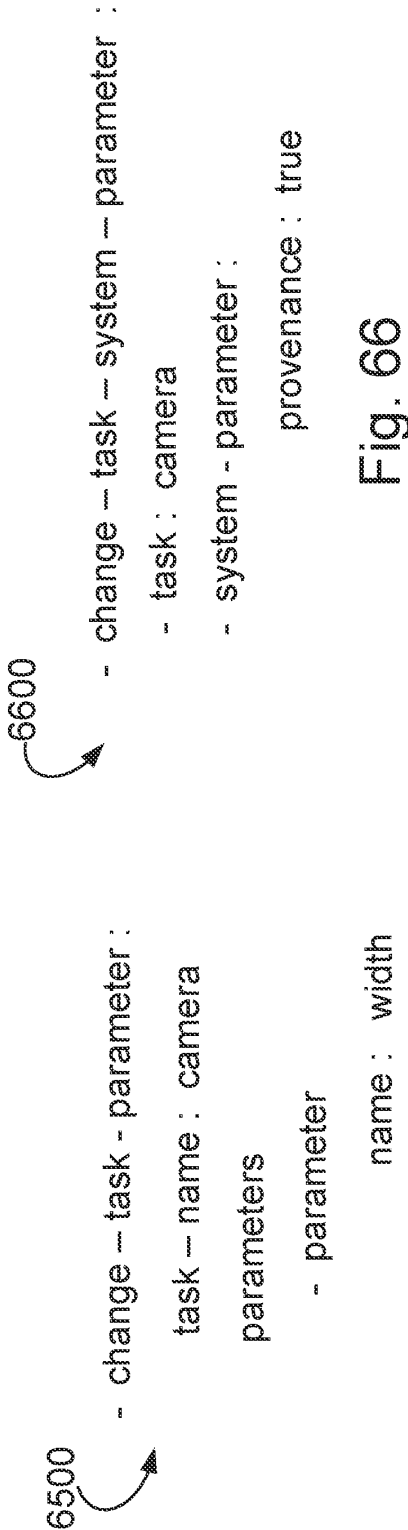


Fig. 63

Fig. 64



38/45

- change-operator-implementetation:

operator-name: CameraPiSensor

implementation:

language: python

module: demo . tasks. camnera_sensor_pi_camera

pip :

- "picamera==1.13"

- "Pillow==8.3.22"

6900



Fig. 69

7000



add-operator :

operator : building

port : new_porf

kind : "process"

Fig. 70

7100



remove-operator :

operator : building

port : deleted_port

Fig. 71

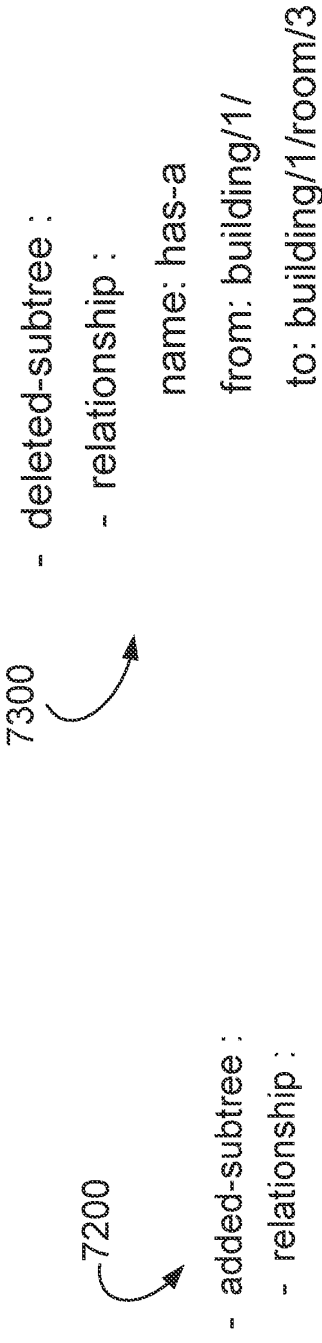


Fig. 73

Fig. 72

40/45

7400

- change-task-instance-parameter:
task-name: camera
Instance_id: building/1/room/1/door/1/
parameters
-parameters
name: width
value: 640

Fig. 74

7500

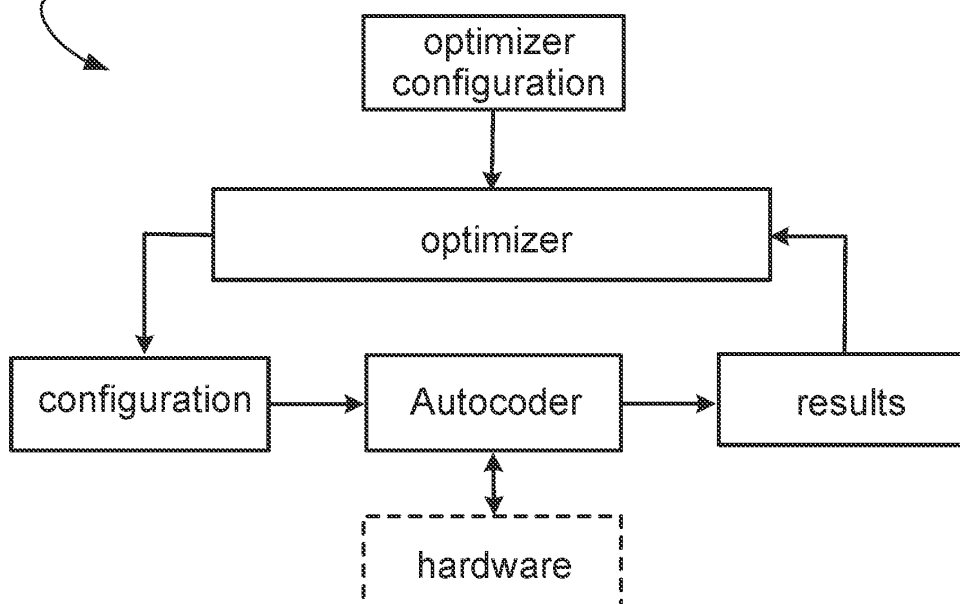
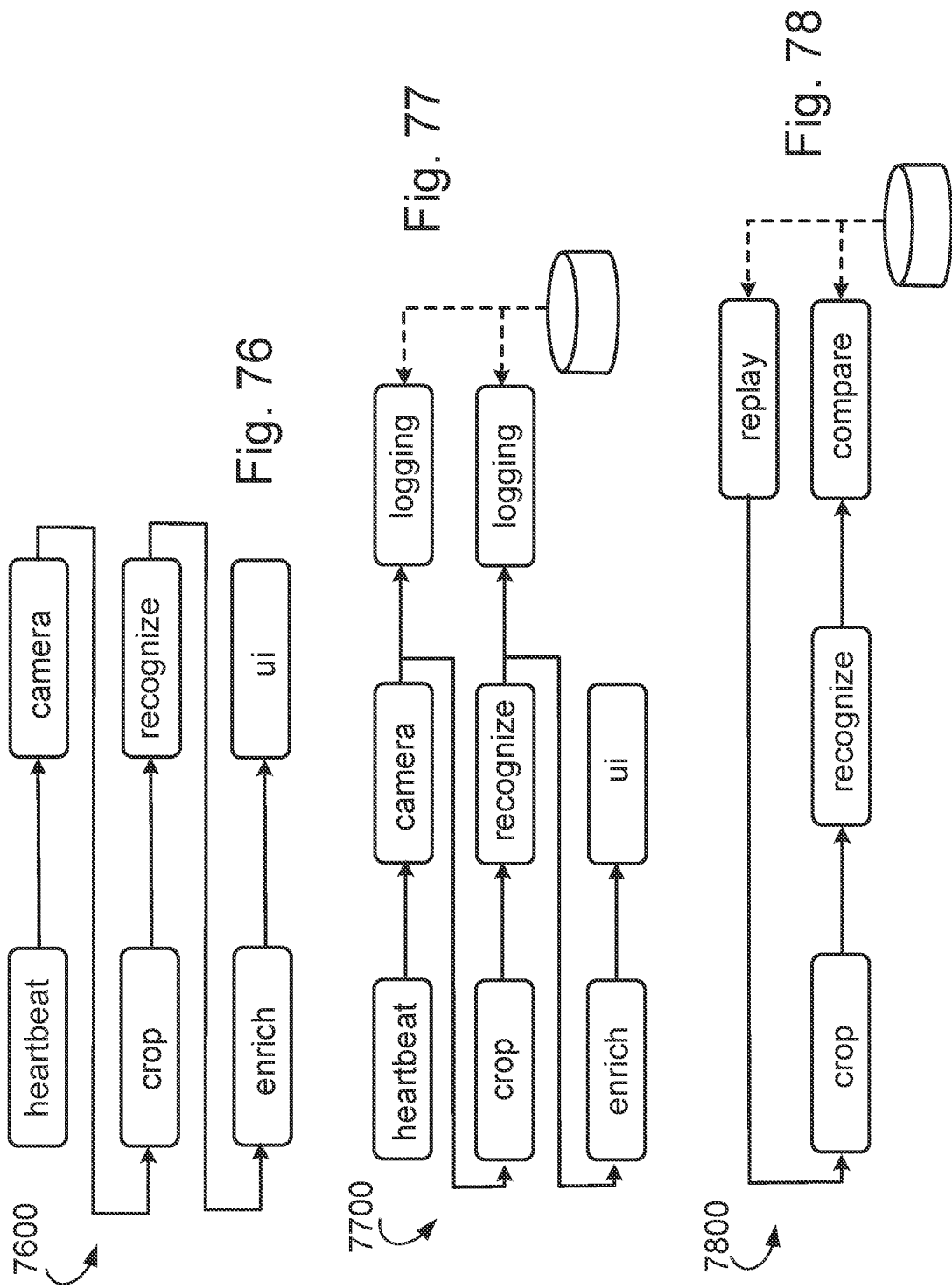


Fig. 75



42/45

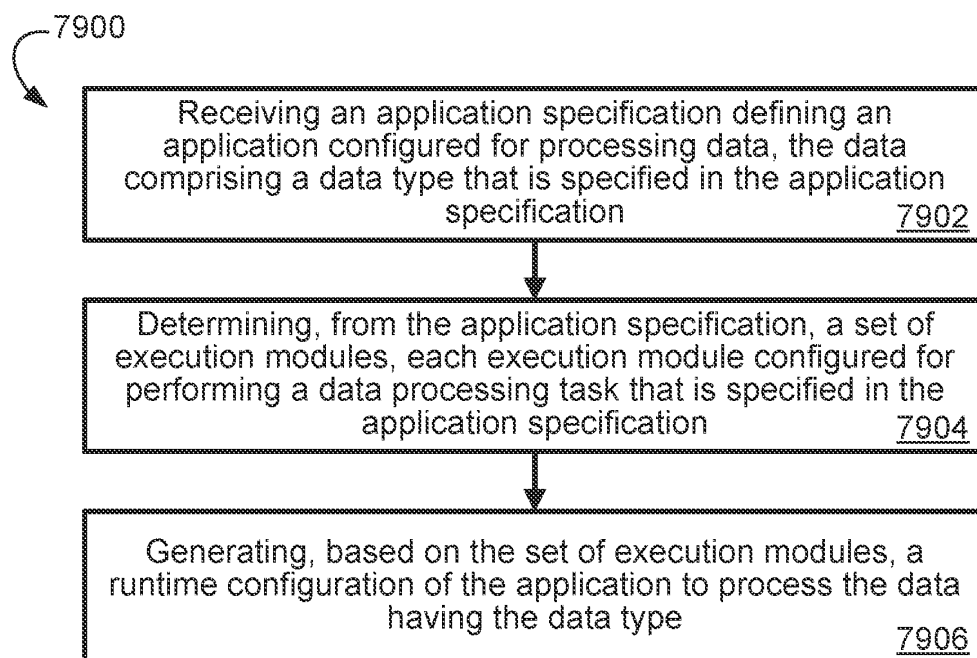


Fig. 79

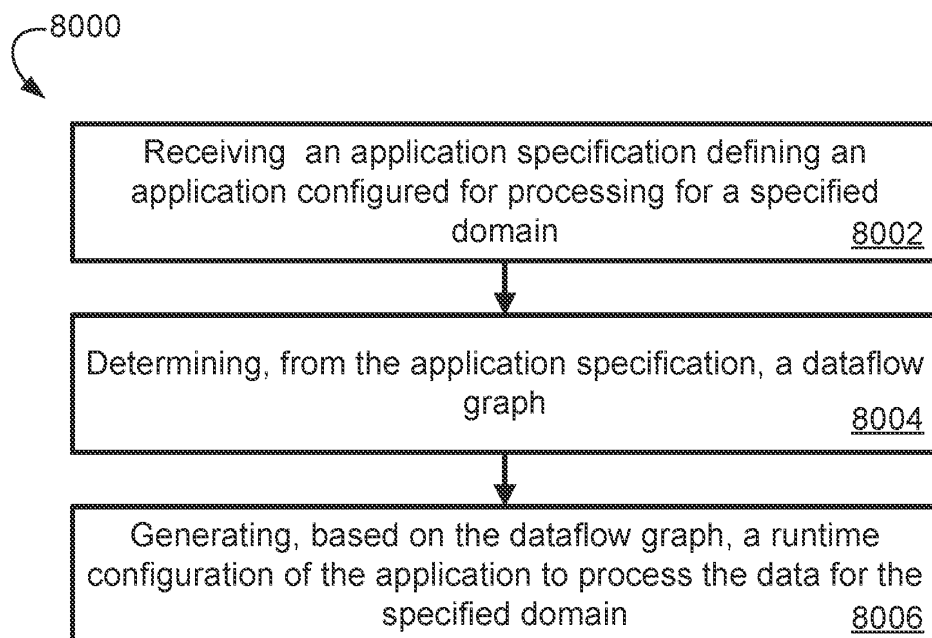


Fig. 80

43/45

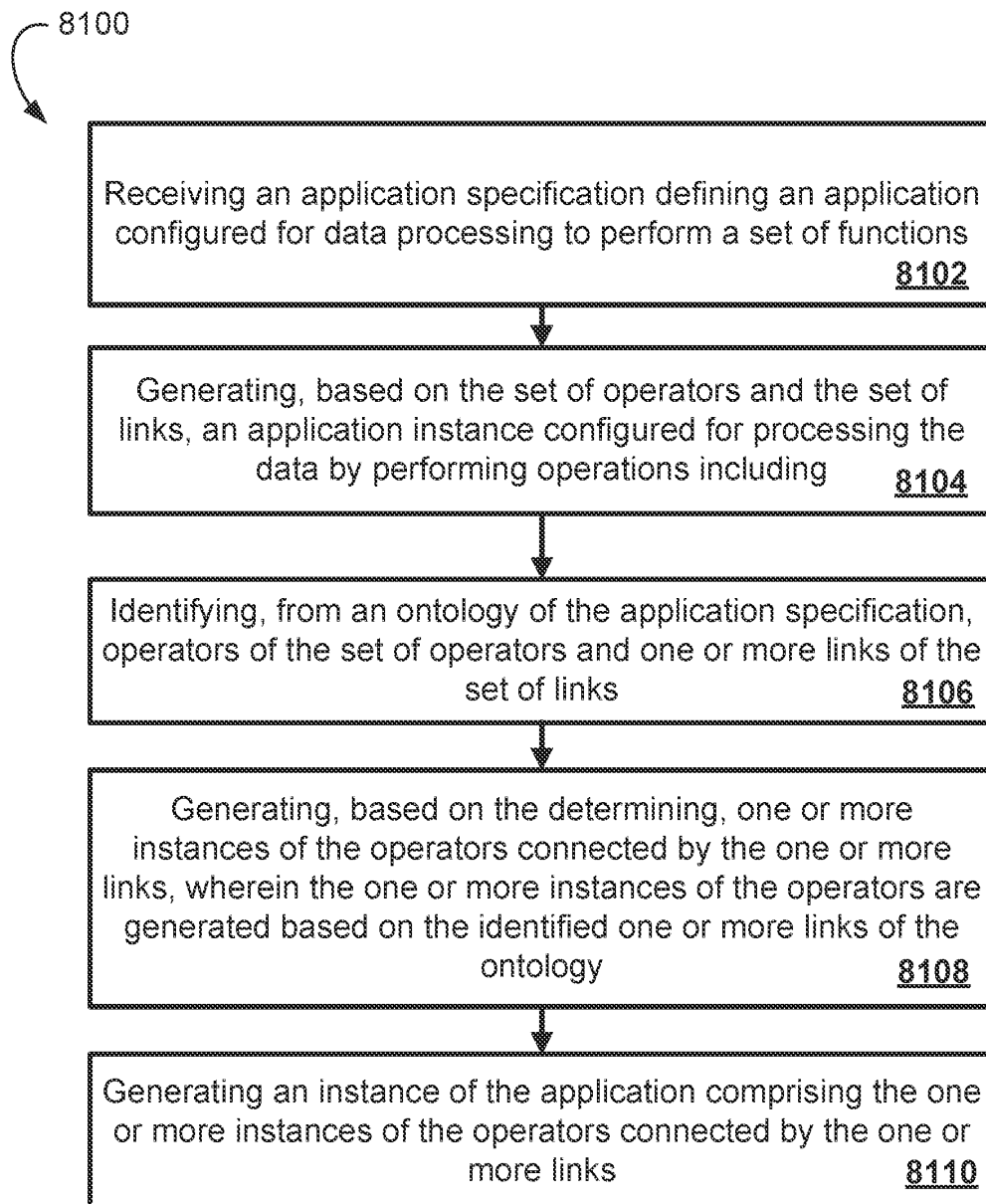


Fig. 81

44/45

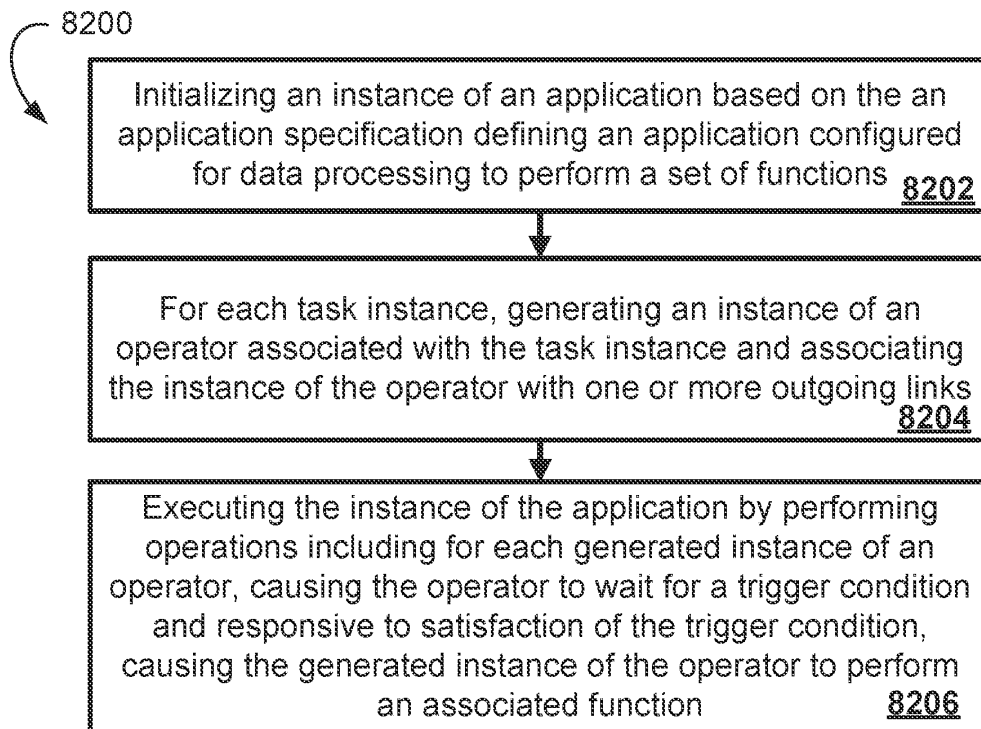


Fig. 82

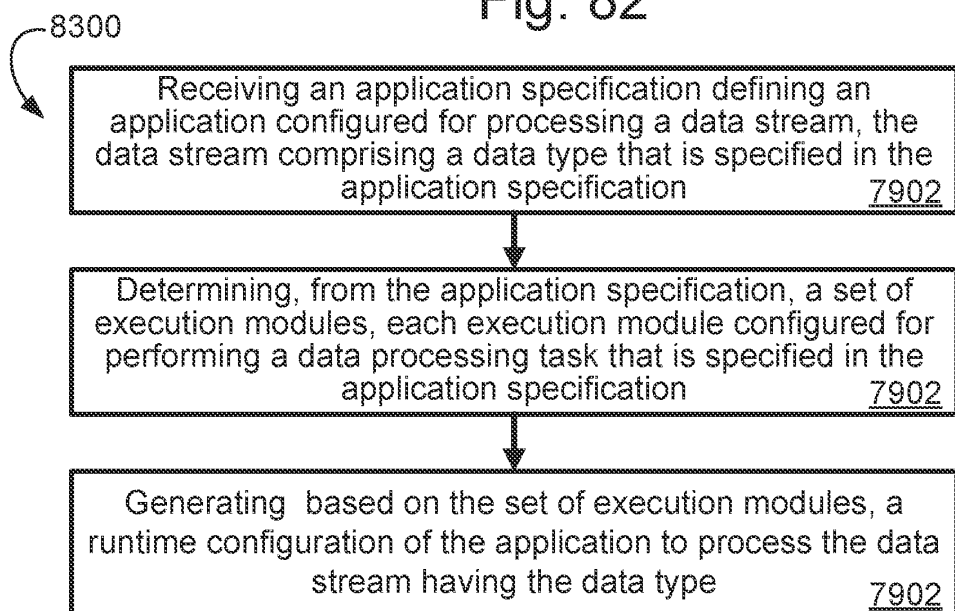


Fig. 83

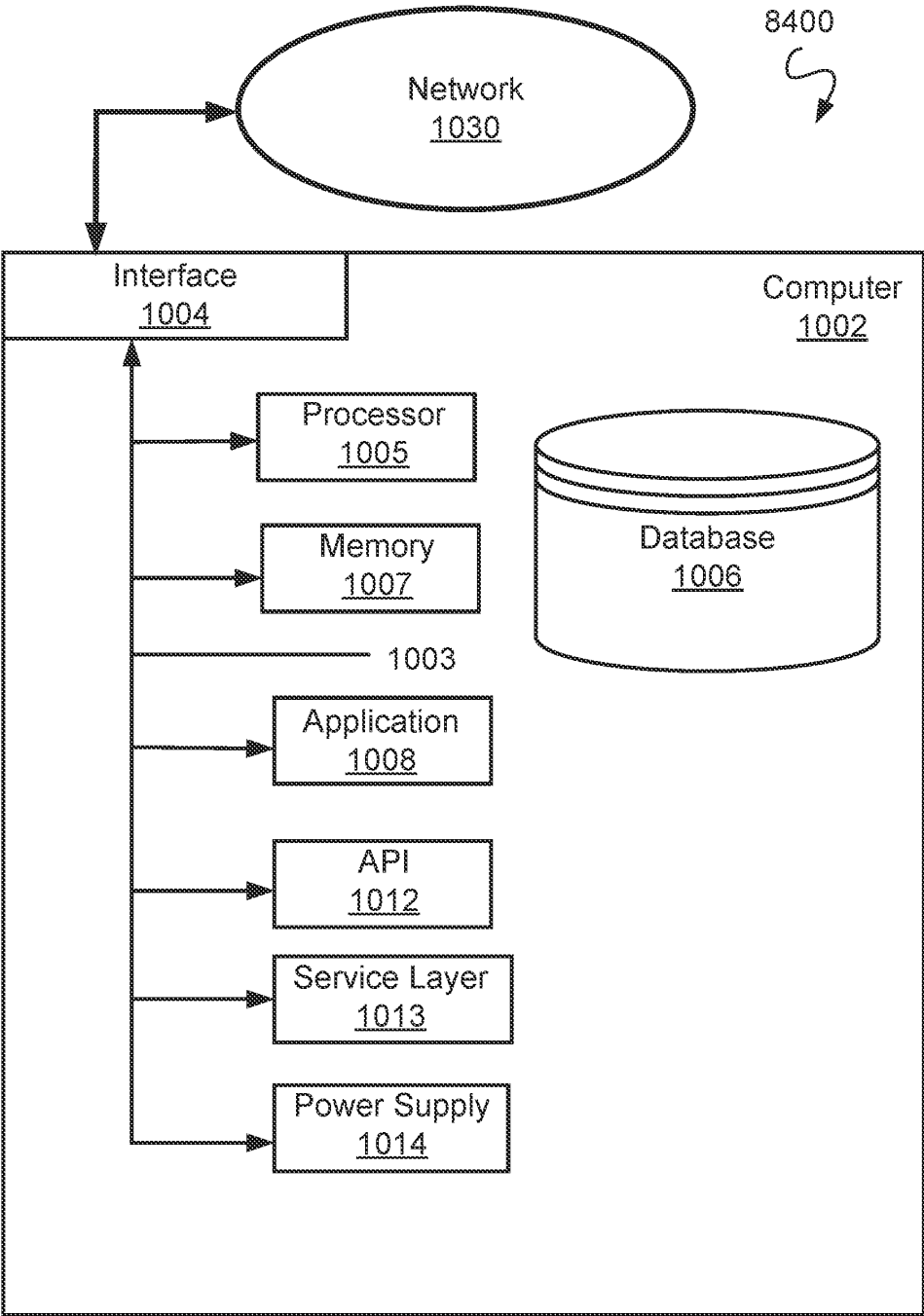


FIG. 84