■ 付録 1 実験で使用した ReAct-Dac ソースコード全体

ベースライン（1 回完結方式）は，当該ソースコードからレビューノード（87 行目）を削除して実行した.

```
import os, sys, subprocess, webbrowser
import json
import base64
import shutil
import datetime
import time
from pathlib import Path
from typing import Annotated, Literal
from typing_extensions import TypedDict
from pydantic import BaseModel, Field
import gradio as gr

from operator import add
from langgraph.graph import StateGraph, START, END
from langgraph.checkpoint.memory import InMemorySaver
from langchain_openai import ChatOpenAI
from langchain_core.messages import BaseMessage, SystemMessage, HumanMessage, AIMessage
from openai import OpenAI

class RequirementList(BaseModel):
    req_list: list[str] = Field(
        ...,
        description="Ordered REQ list like 'REQ-01: ...'.",
    )

class SWInput(BaseModel):
    sw_input: str = Field(..., description="PlantUML code block")

class Review(BaseModel):
    review_passed: bool = Field(..., description="Whether the generated PlantUML diagram passed the review")
    improvement_points: list[str] = Field(..., description="List of improvement points if the review did not pass")

class State(TypedDict):
    messages: Annotated[list[BaseMessage], add]
    requirements_txt: str
    requirement_list: list[str]
    attempt_num: int
    sw_input_gen: bool
    sw_input: str
    sw_exec: str
    sw_output: str
    sw_stderr: str
    review_gen: bool
    review_passed: bool
    improvement_points: list[str]

class PlantUMLAgent:
    def __init__(self, plantuml_path: str = "./plantuml", max_iterations: int = 3, use_web: bool = False):
        print("Agent initializing...")
        api_key = os.getenv("OPENAI_API_KEY")
        if not api_key:
            raise ValueError("OPENAI_API_KEY environment variable is required")

        self.oclient = OpenAI()
        #       self.llm      =      ChatOpenAI(model="gpt-4.1-2025-04-14",      temperature=0.2, output_version="responses/v1")
        self.llm    =    ChatOpenAI(model="gpt-5.2-2025-12-11",    reasoning={"effort":    "medium"}, output_version="responses/v1") # none / low / medium / high / xhigh
        self.llm_response_requirement_list = self.llm.with_structured_output(RequirementList)
        self.llm_response_sw_input = self.llm.with_structured_output(SWInput)
        self.llm_response_review = self.llm.with_structured_output(Review)

        self.use_web = use_web
        self.tools = [{"type":"web_search"}] if use_web else []
        self.plantuml_path = Path(plantuml_path).resolve()
        self.max_iterations = max_iterations
        self.tmp_dir = Path("./tmp").resolve()
        self.tmp_dir.mkdir(exist_ok=True)

        ts = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")
        self.log_file = self.tmp_dir / f"plantuml_agent_log_{ts}.json"

        self.checkpointer = InMemorySaver()
        self.workflow = self._build_workflow()
        self.app = self.workflow.compile(checkpointer=self.checkpointer)

        try:
            out_path = self.tmp_dir / "workflow_graph.png"
            self.app.get_graph().draw_mermaid_png(output_file_path=str(out_path))
            self._open_image(out_path)
        except Exception as e:
            print(f"Error generating workflow graph: {e}")

    def _build_workflow(self) -> StateGraph:
        g = StateGraph(State)
        g.add_node("build_software", self.build_plantuml_if_needed)
        g.add_node("generate_input", self.generate_plantuml_code)
        g.add_node("execute_software", self.execute_plantuml)
        g.add_node("review_output", self.review_plantuml_diagram) #
```

```
 88
 89          g.add_edge(START, "build_software")
 90          g.add_edge("build_software", "generate_input")
 91          g.add_conditional_edges(
 92              "generate_input",
 93              self.should_retry_generation,
 94              {
 95                  "failed": "generate_input",
 96                  "succeeded": "execute_software",
 97                  "max_attempts_reached": END
 98              },
 99          )
100          g.add_edge("execute_software", "review_output") #
101          g.add_conditional_edges( #
102              "review_output",
103              self.should_retry,
104              {
105                  "needs_improvement": "generate_input",
106                  "passed_OR_max_attempts_reached": END
107              }
108          )
109          return g
110
111      # --- nodes ---
112      def build_plantuml_if_needed(self, state: State) -> dict:
113          print("Checking if PlantUML JAR needs to be built...")
114          build_libs_dir = self.plantuml_path / "build" / "libs"
115          jar = self._find_plantuml_jar(build_libs_dir)
116          if not jar:
117              try:
118                  gradle_cmd = str((self.plantuml_path / ("gradlew.bat" if sys.platform == "win32"
    else "gradlew")).resolve())
119                  subprocess.run([gradle_cmd,  "build",  "-x",  "test"],  cwd=self.plantuml_path,
    check=True, timeout=600)
120              except Exception as e:
121                  print(f"Error building PlantUML: {e}")
122              jar = self._find_plantuml_jar(build_libs_dir)
123
124          if not jar:
125              raise FileNotFoundError(f"PlantUML JAR file not found in {self.plantuml_path}")
126          return  {"messages":  [SystemMessage(content=f"[OBSERVATION]  PlantUML  JAR:  {jar}")],
    "sw_exec": jar}
127
128      def generate_plantuml_code(self, state: State) -> dict:
129          attempt = state.get("attempt_num", 0) + 1
130          print(f"¥n🚀 Attempt {attempt} to generate PlantUML code...")
131
132          system_msg = SystemMessage(content="あなたは PlantUML 図の生成者です。要求に基づいて正確な PlantUML
    コードを生成してください。")
133          prompt = f"""¥
134  以下の要求に基づいて、動作する PlantUML コードを生成してください。
135  前回の PlantUML コードに対する改善点があれば、それを反映したコードを生成してください。
136
137  要求: {state['requirements_txt']}
138
139  前回の PlantUML コードに対する改善点:
140  {chr(10).join(state.get('improvement_points', [])) if state.get('improvement_points') else 'なし
    '}
141
142  REQ 一覧:
143  {chr(10).join(state.get('requirement_list', [])) if state.get('requirement_list') else '未生成'}
144
145  以下の手順で生成してください。
146
147  生成手順:
148  0) ルール
149     - もし、生成済みの REQ 一覧が存在する場合は、それをそのまま再利用し、1)要求分解 2)過剰要素を含めないための要求
    追加 は実行しない。
150     - 未生成の場合のみ、1)と 2)を実行する。
151
152  1) 要求分解
153     - 元の要求文と添付ファイルから、満たすべき事項を検証可能な粒度に分解し、REQ-01, REQ-02... の ID を付与する。
154     - REQ-00 は「図のレンダリングが成功していること」とし、必ず REQ 一覧の最初に含める。
155     - 「〜を含めない」「〜をしない」も同様に要求（REQ）として扱い、REQ 一覧に含める。
156
157  2) 過剰要素を含めないための要求追加（連番を継続）
158     - この手順の目的は、期待要素の全量を定義し、それ以外の要素が図に存在しないことを要求として明文化し、REQ 一覧に
    追加することである。
159     - 追加する要求は、別リストにせず、手順 1 で作成した REQ-01, REQ-02... の末尾に、連番を継続して追記すること（例：
    REQ-05 まで作ったなら、次は REQ-06 として追加）。
160     - 元の要求文と添付ファイルから過剰要素を特定できない場合、この手順をスキップすること。
161       - 例：「図書館システムのクラス図を作成してください」→ 要求が曖昧で期待要素を確定できないため、逆に過剰要素を特
    定できない。
162     - 過剰要素を含めない要求の作成方法と要求の例は以下の通り。
163       - 作成方法（期待要素の列挙＋差分探索）：期待される要素（クラス/関係/ラベル等）をリスト化し、そのリストに含まれ
    ない要素を図に含めないことを 1 つの要求として作成し、REQ 一覧の末尾に追記する。
164       - 要求例「REQ-06: 顧客、注文、注文明細、商品以外のクラスが存在しないこと」
165
166  3) REQ 一覧の全てを満たし、前回の PlantUML コードに対する改善点を反映した、動作する PlantUML コードを生成する
167
168  4) 出力への落とし込み（フォーマット厳守）
169     - 「要求一覧」には、生成した REQ 一覧を列挙する。
170     - 「PlantUML コード」には、生成した PlantUML コードを書く
171     - 「簡易説明」には、PlantUML コードの簡易説明または改善箇所の説明を書く
172
173  望ましい出力形式:
174  要求一覧: [REQ 一覧]
175  PlantUML コード: [PlantUML コード]
176  簡易説明: [PlantUML コードの簡易説明または改善箇所の説明]
```

```
177 """
178         user_msg = HumanMessage(content=prompt)
179 
180         try:
181             ai                          =                          self.llm.bind_tools(self.tools,
    tool_choice="auto").invoke(self._keep_only_last_image(state["messages"]      +      [system_msg,
    user_msg]))
182             self._log_web_search_results(ai)
183             req_list                =                state.get("requirement_list")               or
    self.llm_response_requirement_list.invoke([ai]).req_list
184             sw_input = self.llm_response_sw_input.invoke([ai]).sw_input
185             return   {"messages":   [system_msg,   user_msg,   ai],   "requirement_list":   req_list,
    "sw_input": sw_input, "attempt_num": attempt, "sw_input_gen": True}
186         except Exception as e:
187             print(f"PlantUML code generation error: {e}")
188             return {
189                 "messages":  [user_msg,  SystemMessage(content=f"[SYSTEM  ERROR]  PlantUML 生成失敗:
    {e}")],
190                 "sw_input": "",
191                 "attempt_num": attempt,
192                 "sw_input_gen": False,
193             }
194 
195     def   should_retry_generation(self,   state:   State)   ->   Literal["failed",   "succeeded",
    "max_attempts_reached"]:
196         attempt_num = state.get("attempt_num", 0)
197         sw_input_gen = state.get("sw_input_gen", False)
198 
199         if attempt_num >= self.max_iterations:
200             return "max_attempts_reached" if not sw_input_gen else "succeeded"
201 
202         return "failed" if not sw_input_gen else "succeeded"
203 
204     def execute_plantuml(self, state: State) -> dict:
205         print("Executing PlantUML...")
206         code, jar = state.get("sw_input", ""), state.get("sw_exec", "")
207         ts = datetime.datetime.now().strftime("%Y%m%d_%H%M%S_%f")[:-3]
208         puml = self.tmp_dir / f"plantuml_diagram_{ts}.puml"
209         png = self.tmp_dir / f"plantuml_diagram_{ts}.png"
210         puml.write_text(code, encoding="utf-8")
211 
212         try:
213             r = subprocess.run(["java", "-DPLANTUML_LIMIT_SIZE=8192", "-jar", jar, "-stdrpt:2", "-
    tpng", str(puml)], capture_output=True, text=True, timeout=30)
214             temp_png = puml.with_suffix(".png")
215             if temp_png.exists():
216                 shutil.move(str(temp_png), str(png))
217                 self._open_image(png)
218                 msg = f"PlantUML 実行成功: {png.name}"
219                 ok = True
220             else:
221                 msg = f"PlantUML 実行失敗: {r.stderr.strip()}"
222                 ok = False
223         except Exception as e:
224             print(f"PlantUML execution error: {e}")
225             msg, ok = f"PlantUML 実行エラー: {e}", False
226 
227         return {"messages": [SystemMessage(content=f"[OBSERVATION] {msg}")], "sw_output": str(png
    if ok else ""), "sw_stderr": (r.stderr or "")}
228 
229     def review_plantuml_diagram(self, state: State) -> dict:
230         print("Generating review...")
231         out = state.get("sw_output", "")
232         if not out:
233             obs = "画像未生成"
234             return  {"messages":  [SystemMessage(content=f"[OBSERVATION]  {obs}")],  "review_gen":
    False, "review_passed": False}
235 
236         try:
237             b64 = base64.b64encode(Path(out).read_bytes()).decode("utf-8")
238         except Exception as e:
239             print(f"Image read error: {e}")
240             obs = f"画像読込失敗: {e}"
241             return  {"messages":  [SystemMessage(content=f"[OBSERVATION]  {obs}")],  "review_gen":
    False, "review_passed": False}
242 
243         system_msg = SystemMessage(content="あなたは PlantUML 図の評価者です。要求適合性を評価してください。
    ")
244         use_web_prompt = "Web 検索ツールを使用して情報を取得し、" if self.use_web else ""
245         review_prompt = f"""¥
246 要求に対して、以下の PlantUML 図を生成しました。
247 元の要求: {state['requirements_txt']}
248 
249 要求一覧:
250 {chr(10).join(state.get('requirement_list', [])) if state.get('requirement_list') else 'なし'}
251 
252 生成した PlantUML 図: [画像を参照]
253 
254 PlantUML 実行時の標準エラー出力: {state.get('sw_stderr', '').strip() or 'なし'}
255 
256 以下の手順で評価してください。
257 この回は前回の判定・修正内容を一切信用せず、提示された“今回の画像”だけで再判定する。前回 YES/NO は参照禁止。
258 
259 評価手順:
260 0) 検証実行（REQ-00 のみ判定）
261    - REQ-00 が NO なら、以降の REQ 判定は行わず「要改善」で確定（他 REQ は未評価でよい）
262    - PlantUML 実行時の標準エラー出力にエラーが含まれている場合、REQ-00 は NO とする。
263 
264 1) 検証実行（REQ-01 以降のチェック）
```

```
265     - REQ 一覧の各項目について、「生成した PlantUML 図」を根拠に YES / NO を判定する。
266     - 判定ごとに、図から読み取れる根拠を 1 行で添える（例：「図中に『A』と『B』があり、矢印で接続」）。
267
268   2) 評価判定（満足/要改善）
269     - 満足：すべての REQ が YES
270     - 要改善：いずれかの REQ が NO
271
272   3) 改善点の特定
273     - 要改善の場合、NO になった REQ ごとに、{use_web_prompt}PlantUML コードの修正内容を特定する。
274
275   4) 出力への落とし込み（フォーマット厳守）
276     - 「原因」には、NO になった REQ のみを、REQ 番号付きで列挙する（各項目に短い根拠も添える）。
277     - 「PlantUML コードの改善点」には、原因の各項目ごとに修正内容を詳細に書く。
278     - 「付録」には、全 REQ の検証結果（REQ 番号 / YES・NO / 根拠）を一覧で列挙する。
279
280   望ましい出力形式:
281   評価: [満足/要改善]
282   原因: [NO になった REQ のみを根拠を添えて列挙]
283   PlantUML コードの改善点: [原因の各項目に対して、PlantUML コードの修正内容を可能な限り詳細に説明]
284   付録: [検証結果]
285   """
286           analysis = HumanMessage(content=[{"type": "input_text", "text": review_prompt}, {"type":
      "input_image", "image_url": f"{b64}"}])
287           tool_choice = "web_search" if self.use_web else "auto"
288
289           try:
290               ai                              =                             self.llm.bind_tools(self.tools,
      tool_choice=tool_choice).invoke(self._keep_only_last_image(state["messages"]    +    [system_msg,
      analysis]))
291               self._log_web_search_results(ai)
292               review = self.llm_response_review.invoke([ai])
293               print("Improvement points:¥n" + "¥n".join(f"- {p}" for p in (review.improvement_points
      or ["なし"])))
294               return {"messages": [system_msg, analysis, ai], "review_gen": True, "review_passed":
      review.review_passed, "improvement_points": review.improvement_points}
295           except Exception as e:
296               print(f"Review generation error: {e}")
297               obs = f"レビュー失敗: {e}"
298               return   {"messages":   [system_msg,   analysis,   SystemMessage(content=f"[OBSERVATION]
      {obs}")], "review_gen": False, "review_passed": False}
299
300       def      should_retry(self,      state:      State)      ->      Literal["needs_improvement",
      "passed_OR_max_attempts_reached"]:
301           if state.get("review_passed", False):
302               return "passed_OR_max_attempts_reached"
303           if state.get("attempt_num", 0) >= self.max_iterations:
304               return "passed_OR_max_attempts_reached"
305           return "needs_improvement"
306
307       # --- utils ---
308       def _find_plantuml_jar(self, build_libs_dir: Path) -> str:
309           if not build_libs_dir.exists():
310               return ""
311           jars = sorted(
312               [p for p in build_libs_dir.glob("plantuml*.jar") if not (str(p).endswith("-sources.jar")
      or str(p).endswith("-javadoc.jar"))],
313               key=lambda p: p.name,
314               reverse=True,
315           )
316           return str(jars[0]) if jars else ""
317
318       def _open_image(self, p: Path):
319         p = str(Path(p).resolve())
320         try:
321             {"win32": lambda x: os.startfile(x),
322             "darwin": lambda x: subprocess.run(["open", x], check=False)
323             }.get(sys.platform, lambda x: subprocess.run(["xdg-open", x], check=False))(p)
324         except Exception as e:
325             try:
326                 webbrowser.open(Path(p).as_uri())
327             except Exception as e2:
328                 print(f"[WARN] open failed: {e} / browser: {e2}")
329
330       def _keep_only_last_image(self, msgs: list[BaseMessage]) -> list[BaseMessage]:
331           found, out = False, []
332           for m in reversed(msgs):
333               if not isinstance(m.content, list):
334                   out.append(m)
335                   continue
336               parts = []
337               for p in m.content:
338                   if isinstance(p, dict) and p.get("type") == "input_image":
339                       p = p if not found else {"type": "input_text", "text": "[image removed]"}
340                       found = True
341                   parts.append(p)
342               out.append(m.__class__(content=parts, name=getattr(m, "name", None)))
343           return list(reversed(out))
344
345       # --- logging ---
346       def _log_web_search_results(self, ai: AIMessage):
347           c = ai.content
348           qs = [p['action']['query'] for p in c if p['type'] == 'web_search_call']
349           cs = [(a['title'], a['url']) for p in c for a in p.get('annotations', [])]
350           if qs or cs:
351               print("¥n".join((["Web  Queries:"]+[f"-  {q}"  for  q  in  qs]  if  qs  else  [])  +
      (["Citations:"]+[f"- {t} | {u}" for t,u in cs] if cs else [])))
352
353       def _save_messages_to_log_file(self, result: dict, thread_id: str):
354           data = {
```

```
            "agent_started_at": self.started_at,
            "agent_finished_at": float(int(time.time())),
            "thread_id": thread_id,
            "messages":    [{"type":    (d:=m.model_dump()).pop("type"),    **d}    for    m    in
result.get("messages", [])],
            "final_state": {
                "review_gen": result.get("review_gen", False),
                "review_passed": result.get("review_passed", False),
                "attempt_num": result.get("attempt_num", 0),
                "sw_output": result.get("sw_output", ""),
                "sw_input_gen": result.get("sw_input_gen", False),
            },
        }
        try:
            self.log_file.write_text(json.dumps(data, ensure_ascii=False, indent=2), encoding="utf-
8")
        except Exception as e:
            print(f"Error saving log file: {e}")

    # --- API ---
    def invoke(
        self,
        requirements_txt: str,
        thread_id: str = "default"
    ) -> str:
        self.started_at = float(int(time.time()))
        config = {"configurable": {"thread_id": thread_id}}
        init = {
            "messages": [HumanMessage(content=requirements_txt)],
            "requirements_txt": requirements_txt,
            "requirement_list": [],
            "attempt_num": 0,
            "sw_input": "",
            "sw_output": "",
            "sw_stderr": "",
            "review_gen": False,
            "review_passed": False,
            "sw_input_gen": False,
            "sw_exec": "",
        }

        try:
            result = self.app.invoke(init, config)
            self._save_messages_to_log_file(result, thread_id)
        except Exception as e:
            return f"Error occurred: {e}"

        if result.get("review_passed"):
            return "Review passed."

        attempt = result.get("attempt_num", 0)
        if not result.get("sw_input_gen", False):
            return "PlantUML Code generation failed."
        if not result.get("review_gen", False):
            return "Review generation failed."
        if attempt >= self.max_iterations:
            msg = f"{attempt} attempts made but failed to generate a valid diagram."
            return msg
        return "PlantUML code generation failed. Please try again."

def main():
    plantuml_path = input("Path to PlantUML project [./plantuml]: ").strip() or "./plantuml"
    try:
        max_iterations = int(input("Maximum code generation attempts [5]: ").strip() or "5")
    except ValueError:
        max_iterations = 3
    use_web = (input("Use web search? [Y/n]: ").strip().lower() or "y") in {"y", "yes"}

    agent = PlantUMLAgent(plantuml_path, max_iterations, use_web)
    thread_id = "plantuml_session"

    def run(requirements_txt):
        res = agent.invoke(requirements_txt or "", thread_id)
        print(f"¥n{res}")
        return res

    gr.Interface(
        fn=run,
        inputs=[
            gr.Textbox(lines=12, label="要求"),
        ],
        outputs=[gr.Textbox(lines=1, label="結果")],
        title="PlantUML Agent", flagging_mode="never"
    ).launch()

if __name__ == "__main__":
    main()
```

■　付録 2 実験で実行したタスク一覧

タスク名	内容
CLS_M	図種：クラス図 # クラス 以下の 12 クラスのみを用いる。クラス名は完全一致とする。 学生

	講座 講師 講座回 受講登録 請求 支払 割引クーポン 教材 進捗 修了証 通知 このうち abstract なクラスを明示する。abstract とするクラスは「通知」とする。 インターフェースの実装（realization）を含める。インターフェースとして「送信可能」を追加し、通知がこれを実装すること。 ※ただし、新規に追加できる要素はインターフェース「送信可能」のみとし、既存の 12 クラスは増減しない。 # 関連 以下に列挙した関連のみを引く。関連名を付けるものは、指定どおりの関連名を用いる。 関連名を付ける場合は、関連名の中に、関連名を読む際の主語を示すベタ塗りの三角形（UML 2.5.1 における“the solid triangle”）を 1 つ含めること。三角形は、底辺側に位置するクラスが主語となる向きで配置すること。 1. 学生 − 受講登録（関連名：登録する、主語：学生） 2. 講座 − 講座回（関連名：含む、主語：講座） 3. 講師 − 講座（関連名：担当する、主語：講師） 4. 受講登録 − 講座回（関連名なし） 5. 受講登録 − 請求（関連名：請求される、主語：受講登録） 6. 請求 − 支払（関連名：精算する、主語：請求） 7. 受講登録 − 割引クーポン（関連名：適用する、主語：受講登録） 8. 講座回 − 教材（関連名：提供する、主語：講座回） 9. 受講登録 − 進捗（関連名：記録する、主語：受講登録） 10. 受講登録 − 修了証（関連名：発行する、主語：受講登録） 11. 学生 − 通知（関連名：受け取る、主語：学生） # 多重度 各関連の両端に多重度を設定する。以下のとおりとする。 1. 学生(0..*) − 受講登録(1) 2. 講座(1..*) − 講座回(1) 3. 講師(0..*) − 講座(1..*) 4. 受講登録(1) − 講座回(0..*) 5. 受講登録(0..1) − 請求(1) 6. 請求(1) − 支払(1..*) 7. 受講登録(0..1) − 割引クーポン(0..*) 8. 講座回(1..*) − 教材(1) 9. 受講登録(0..*) − 進捗(1) 10. 受講登録(0..1) − 修了証(1) 11. 学生(0..*) − 通知(1) # 可視性 すべての属性と操作に可視性を付与する。 属性はすべて private とする。 操作はすべて public とする。 # 属性 各クラスに以下の属性を持たせる。各属性には型を付ける。 また、abstract な属性を明示する。abstract 属性は「通知.本文:String」とする。 また、static な属性を明示する。static 属性は「割引クーポン.最大割引率:Percentage」とする。 学生：学生 ID:String、氏名:String、メールアドレス:String、在籍状態:在籍状態 講座：講座 ID:String、講座名:String、標準受講料:Money 講師：講師 ID:String、氏名:String、メールアドレス:String 講座回：講座回 ID:String、開始日時:DateTime、終了日時:DateTime、開催形式:開催形式、定員:Integer 受講登録：受講登録 ID:String、登録日時:DateTime、状態:受講登録状態、出欠状態:出欠状態 請求：請求 ID:String、発行日:Date、支払期限日:Date、請求総額:Money、状態:請求状態 支払：支払 ID:String、支払日:Date、支払金額:Money、支払手段:支払手段 割引クーポン:クーポンコード:String、割引種別:割引種別、割引値:Decimal、有効開始日:Date、有効終了日:Date 教材：教材 ID:String、教材名:String、提供形態:提供形態、参照先:URI 進捗：進捗 ID:String、記録日時:DateTime、達成率:Percentage 修了証：修了証 ID:String、発行日:Date、証明書番号:String 通知：通知 ID:String、送信日時:DateTime、チャネル:通知チャネル、件名:String # 操作 各クラスに以下の操作を持たせる。各操作には引数と戻り値を付ける。 また、static な操作を明示する。static 操作は「支払.支払を登録する(...)」とする。 学生： 受講登録する(対象講座回:講座回):受講登録 受講登録を取消する(対象受講登録:受講登録):void 通知一覧を確認する():List<通知> 講座： 講座回を作成する(開始日時:DateTime, 終了日時:DateTime, 開催形式:開催形式, 定員:Integer):講座回 講師を割り当てる(担当講師:講師):void

	講師： 講座を担当する(担当講座:講座):void 進捗を評価する(対象進捗:進捗, 評価コメント:String):void 講座回： 教材を追加する(追加教材:教材):void 出欠を記録する(対象受講登録:受講登録, 出欠状態:出欠状態):void 受講登録： クーポンを適用する(クーポン:割引クーポン):void 請求を生成する():請求 進捗を記録する(達成率:Percentage, 記録日時:DateTime):進捗 修了証を申請する():void 請求： 残額を計算する():Money 支払を受領する(支払:支払):void 精算済みにする():void 支払： 支払を登録する(請求:請求, 支払金額:Money, 支払日:Date, 支払手段:支払手段):void 返金する(返金金額:Money):void 割引クーポン： 有効性を判定する(判定日:Date):Boolean 教材： 教材情報を更新する(教材名:String, 提供形態:提供形態, 参照先:URI):void 進捗： 達成率を更新する(達成率:Percentage, 記録日時:DateTime):void 修了証： 発行する(発行日:Date):void 無効化する(理由:String):void 通知： 送信する(宛先:学生, チャネル:通知チャネル, 件名:String, 本文:String):void # ノート ノートは合計で、(a) クラスに付与するノート 2 件 と (b) 関連に付与するノート 2 件 の計 4 件とする。 クラスに付与するノートは指定どおりとし、本文は完全一致とする。 関連に付与するノートは、指定した関連に付与し、本文は完全一致とする。 クラス向けノート 1（受講登録に付与） 受講登録は、学生が特定の講座回に参加する意思を表す記録である。 クラス向けノート 2（支払に付与） 支払は分割を許可する。請求は 1 回以上の支払で精算される。 関連向けノート 1（受講登録 − 請求 に付与） 請求は受講登録ごとに最大 1 件まで生成できる。 関連向けノート 2（請求 − 支払 に付与） 支払は請求に対して 1 回以上紐づく。分割支払を表現するためである。
CLS_M+S （CLS_M に追記）	# 配置場所（Layout） 受講登録を右領域の中央に配置する。 受講登録の周囲に、以下の 5 クラスをそれぞれ近接配置する（受講登録からの関連線が短くなるようにする）。 上側：講座回 右上：請求 右下：支払 下側：進捗 左下：割引クーポン 修了証は進捗の右側に近接配置する。 # 色（Color / Paint） 請求と支払は、クラス枠の中身（背景色）を同じ色にする。 それ以外のクラスは、上記の色とは異なる背景色にする。 # フォント（Typography） 各クラスについて、クラス名の文字サイズは属性・操作の文字サイズより大きくする。 各クラスについて、太字にするのはクラス名のみとし、属性と操作は太字にしない。 # 形状（Shape） 受講登録と請求の関連線を、他の関連線よりも太くする。 # 可視性（Visibility） 教材は、クラス枠に表示される C アイコンを非表示にする（属性区画と操作区画は表示したままにする）。 通知は、属性区画を非表示にする（クラス名と操作区画は表示したままにする）。 進捗は、操作区画を非表示にする（クラス名と属性区画は表示したままにする）。 # インターフェース表記（Interface Notation） インターフェース「送信可能」は、ロリポップ表記（lollipop）で表現する。
SEQ_M	図種：シーケンス図 # 登場人物（ライフライン） 以下の 9 ライフラインのみを用いる。名称は完全一致とする。

	：の後ろに記載するライフラインの種類を表現すること（UML／ロバストネス分析の概念として解釈すること）。 学生：actor 受講登録画面：boundary 受講登録サービス：control クーポンサービス：control 請求サービス：control 支払サービス：control 進捗サービス：control 修了証サービス：control 通知サービス：control # メッセージ 以下の同期メッセージを記載順どおりに上から下へ配置する。メッセージ名は完全一致とする。 応答メッセージを記載する。内容は対応する同期メッセージの戻り値の型名とすること。 インデントに合わせて呼び出しをネストさせること。 1.学生 → 受講登録画面：受講登録画面表示(学生 ID, セッション ID):メッセージ 2.学生 → 受講登録画面：受講登録する(受講登録, セッション ID):メッセージ 3.受講登録画面 → 受講登録サービス：受講登録作成(受講登録, セッション ID):受講登録 ID 4.受講登録サービス → クーポンサービス：クーポン検証(クーポンコード, 学生 ID):検証結果 # 条件分岐（クーポン検証結果） クーポン検証の結果によって、以下の 2 分岐を表現する。 * 分岐 A（ガード：検証結果が有効） 5.受講登録サービス → 受講登録サービス：クーポン適用(受講登録 ID, クーポンコード):void * 分岐 B（ガード：検証結果が無効） （この分岐では何も呼び出さない） # 請求生成 6.受講登録サービス → 請求サービス：請求生成(受講登録 ID):請求 ID # 分割支払ループ 分割支払を表現するため、繰り返し条件を明示する。 * loop（残額>0） 7.学生 → 受講登録画面：支払実行(請求 ID, 金額):メッセージ 8.受講登録画面 → 支払サービス：支払登録(請求 ID, 金額):支払 ID 9.支払サービス → 請求サービス：支払反映(請求 ID, 支払 ID):残額 # 進捗記録 10.学生 → 受講登録画面：進捗更新(受講登録 ID, 進捗率):メッセージ 11.受講登録画面 → 進捗サービス：進捗記録(受講登録 ID, 進捗率):進捗 ID # 条件分岐（修了） 進捗率が 100%に到達した場合のみ修了証を発行する。 12.学生 → 受講登録画面：修了証発行(受講登録 ID):修了証 PDF 13.受講登録画面 → 修了証サービス：修了証発行(学生 ID, 受講登録 ID):修了証 URL 14.修了証サービス → 進捗サービス：進捗取得(学生 ID, 受講登録 ID):進捗率 * 分岐 C（進捗率=100） 15.修了証サービス → 修了証サービス：修了証発行(学生 ID, 受講登録 ID):修了証 ID 16.修了証サービス → 通知サービス：通知送信(学生 ID, 件名, 本文):通知 ID * 分岐 D（進捗率<100） （この分岐では何も呼び出さない） # イン／アウトメッセージ（図外との送受信） 図の右端へ出ていくアウトメッセージを 1 件追加する（送信元は通知サービス、宛先のライフラインは表示されない）。 17.通知サービス → ：外部通知配信(学生 ID, 件名, 本文):通知 ID 図の左端から受講登録画面へ入ってくるインメッセージを 1 件追加する（宛先は受講登録画面、送信元のライフラインは表示されない）。 18. → 受講登録画面：通知(件名, 本文):void 18 を一番下のメッセージとして表示すること。 # ノート ノートはちょうど 2 件とする。指定したメッセージの近くに付与し、本文は完全一致とする。 ノート 1（メッセージ 6 の近く） クーポンが無効の場合でも請求は生成される。 ノート 2（分割支払いループの近く） 支払は分割を許可するため、同一の請求 ID に対して複数回の支払登録が行われ得る。
SEQ_M+S （SEQ_M に追記）	# 配置場所（Layout） 9 つのライフラインを左から右に、以下の順で配置する。 学生、受講登録画面、受講登録サービス、クーポンサービス、請求サービス、支払サービス、進捗サービス、修了証サービス、通知サービス ライフラインの間隔は均等にし、メッセージ名が重ならない程度の横幅を確保する。 # 色（Color / Paint） 受講登録画面は、ライフラインのヘッダ（名前表示部）を他と異なる背景色にする。 請求サービスと支払サービスは、ライフラインのヘッダ（名前表示部）を同じ背景色にする。 それ以外のライフラインのヘッダは、上記 3 種類の背景色とは異なる背景色にする。 各ライフラインの活性区間には色を付ける。 活性区間が重なっている場合は、色を変える。

	# フォント（Typography） ライフライン名の文字サイズは、メッセージ名の文字サイズより大きくする。 太字にするのはライフライン名のみとし、メッセージ名とガード条件の文字は太字にしない。 # 形状（Shape） 学生（actor）の表記は Awesome man（人物アイコン風）スタイルで表現する。 ライフラインの活性区間を表示する。ネスト（呼び出しの入れ子）に合わせて重なって見えるようにする。 ノートの形状は、六角形にする。 # 可視性（Visibility） ライフラインの下部のボックスは表示しない。

■ 付録 3 チェックリスト一覧

対象タスク名	大カテゴリ	小カテゴリ	チェック内容
CLS_M	クラス	完全一致で存在する	学生
			講座
			講師
			講座回
			受講登録
			請求
			支払
			割引クーポン
			教材
			進捗
			修了証
			通知
			上記以外存在しない
		abstract	通知
		インターフェース	「送信可能」を追加し、通知がこれを実装
	関連		1. 学生 — 受講登録（関連名：登録する、主語：学生）
			2. 講座 — 講座回（関連名：含む、主語：講座）
			3. 講師 — 講座（関連名：担当する、主語：講師）
			4. 受講登録 — 講座回（関連名なし）
			5. 受講登録 — 請求（関連名：請求される、主語：受講登録）
			6. 請求 — 支払（関連名：精算する、主語：請求）
			7. 受講登録 — 割引クーポン（関連名：適用する、主語：受講登録）
			8. 講座回 — 教材（関連名：提供する、主語：講座回）
			9. 受講登録 — 進捗（関連名：記録する、主語：受講登録）
			10. 受講登録 — 修了証（関連名：発行する、主語：受講登録）
			11. 学生 — 通知（関連名：受け取る、主語：学生）
			他の関連が存在しないこと
	多重度		1. 学生(0..*) — 受講登録(1)
			2. 講座(1..*) — 講座回(1)
			3. 講師(0..*) — 講座(1..*)
			4. 受講登録(1) — 講座回(0..*)
			5. 受講登録(0..1) — 請求(1)
			6. 請求(1) — 支払(1..*)
			7. 受講登録(0..1) — 割引クーポン(0..*)
			8. 講座回(1..*) — 教材(1)
			9. 受講登録(0..*) — 進捗(1)
			10. 受講登録(0..1) — 修了証(1)
			11. 学生(0..*) — 通知(1)
	可視性		属性はすべて private とする。
			操作はすべて public とする。
	属性	abstract	abstract 属性は「通知.本文:String」
		static	static 属性は「割引クーポン.最大割引率:Percentage」
		属性	学生：学生 ID:String、氏名:String、メールアドレス:String、在籍状態:在籍状態
			講座：講座 ID:String、講座名:String、標準受講料:Money
			講師：講師 ID:String、氏名:String、メールアドレス:String
			講座回：講座回 ID:String、開始日時:DateTime、終了日時:DateTime、開催形式:開催形式、定員:Integer
			受講登録：受講登録 ID:String、登録日時:DateTime、状態:受講登録状態、出欠状態:出欠状態
			請求：請求 ID:String、発行日:Date、支払期限日:Date、請求総額:Money、状態:請求状態
			支払：支払 ID:String、支払日:Date、支払金額:Money、支払手段:支払手段
			割引クーポン：クーポンコード:String、割引種別:割引種別、割引値:Decimal、有効開始日:Date、有効終了日:Date
			教材：教材 ID:String、教材名:String、提供形態:提供形態、参照先:URI
			進捗：進捗 ID:String、記録日時:DateTime、達成率:Percentage
			修了証：修了証 ID:String、発行日:Date、証明書番号:String
			通知：通知 ID:String、送信日時:DateTime、チャネル:通知チャネル、件名:String
			他の属性が存在しないこと
	操作	static	static 操作は「支払.支払を登録する(...)」
		学生	受講登録する(対象講座回:講座回):受講登録
			受講登録を取消する(対象受講登録:受講登録):void
			通知一覧を確認する():List<通知>
		講座	講座回を作成する(開始日時:DateTime, 終了日時:DateTime, 開催形式:開催

			形式，定員:Integer):講座回
			講師を割り当てる(担当講師:講師):void
		講師	講座を担当する(担当講座:講座):void
			進捗を評価する(対象進捗:進捗，評価コメント:String):void
		講座回	教材を追加する(追加教材:教材):void
			出欠を記録する(対象受講登録:受講登録，出欠状態:出欠状態):void
		受講登録	クーポンを適用する(クーポン:割引クーポン):void
			請求を生成する():請求
			進捗を記録する(達成率:Percentage，記録日時:DateTime):進捗
			修了証を申請する():void
		請求	残額を計算する():Money
			支払を受領する(支払:支払):void
			精算済みにする():void
		支払	支払を登録する(請求:請求，支払金額:Money，支払日:Date，支払手段:支払手段):void
			返金する(返金金額:Money):void
		割引クーポン	有効性を判定する(判定日:Date):Boolean
		教材	教材情報を更新する(教材名:String，提供形態:提供形態，参照先:URI):void
		進捗	達成率を更新する(達成率:Percentage，記録日時:DateTime):void
		修了証	発行する(発行日:Date):void
			無効化する(理由:String):void
		通知	送信する(宛先:学生，チャネル:通知チャネル，件名:String，本文:String):void
		全体	他の操作が存在しないこと
	ノート	クラス向けノート 1（受講登録に付与）	受講登録は、学生が特定の講座回に参加する意思を表す記録である。
		クラス向けノート 2（支払に付与）	支払は分割を許可する。請求は 1 回以上の支払で精算される。
		関連向けノート 1（受講登録 — 請求に付与）	請求は受講登録ごとに最大 1 件まで生成できる。
		関連向けノート 2（請求 — 支払 に付与）	支払は請求に対して 1 回以上紐づく。分割支払を表現するためである。
		全体	他のノートが存在しないこと
CLS_M+S（CLS_M のチェックリストに追加）	配置場所（Layout）	全体	受講登録を右領域の中央に配置する。
		受講登録の周囲に近接配置	上側：講座回
			右上：請求
			右下：支払
			下側：進捗
			左下：割引クーポン
			修了証は進捗の右側に近接配置する。
	色（Color / Paint）		請求と支払は、クラス枠の中身（背景色）を同じ色にする。
			それ以外のクラスは、上記の色とは異なる背景色にする。
	フォント（Typography）		各クラスについて、クラス名の文字サイズは属性・操作の文字サイズより大きくする。
			各クラスについて、太字にするのはクラス名のみとし、属性と操作は太字にしない。
	形状（Shape）		インターフェース「送信可能」は、ロリポップ表記（lollipop）で表現する。
	可視性（Visibility）		教材は、クラス枠に表示される C アイコンを非表示にする（属性区画と操作区画は表示したままにする）。
			通知は、属性区画を非表示にする（クラス名と操作区画は表示したままにする）。
			進捗は、操作区画を非表示にする（クラス名と属性区画は表示したままにする）。
SEQ_M	ライフライン	9 つのライフライン	学生：actor
			受講登録画面：boundary
			受講登録サービス：control
			クーポンサービス：control
			請求サービス：control
			支払サービス：control
			進捗サービス：control
			修了証サービス：control
			通知サービス：control
			上記以外存在しない
	メッセージ	同期メッセージ	1.学生 → 受講登録画面：受講登録画面表示(学生 ID，セッション ID):メッセージ
			2.学生 → 受講登録画面：受講登録する(受講登録，セッション ID):メッセージ
			3.受講登録画面 → 受講登録サービス：受講登録作成(受講登録，セッション ID):受講登録 ID
			4.受講登録サービス → クーポンサービス：クーポン検証(クーポンコード，学生 ID):検証結果
		条件分岐（クーポン検証結果）	* 分岐 A（ガード：検証結果が有効）
			5.受講登録サービス → 受講登録サービス：クーポン適用(受講登録 ID，クーポンコード):void
		条件分岐（クーポン検証結果）	* 分岐 B（ガード：検証結果が無効） （この分岐では何も呼び出さない）

			6.受講登録サービス → 請求サービス：請求生成(受講登録 ID):請求 ID
		分割支払ループ	* loop（残額>0）
			7.学生 → 受講登録画面：支払実行(請求 ID, 金額):メッセージ
			8.受講登録画面 → 支払サービス：支払登録(請求 ID, 金額):支払 ID
			9.支払サービス → 請求サービス：支払反映(請求 ID, 支払 ID):残額
			10.学生 → 受講登録画面：進捗更新(受講登録 ID, 進捗率):メッセージ
			11.受講登録画面 → 進捗サービス：進捗記録(受講登録 ID, 進捗率):進捗 ID
			12.学生 → 受講登録画面：修了証発行(受講登録 ID):修了証 PDF
			13.受講登録画面 → 修了証サービス：修了証発行(学生 ID, 受講登録 ID):修了証 URL
			14.修了証サービス → 進捗サービス：進捗取得(学生 ID, 受講登録 ID):進捗率
		# 条件分岐（修了）	* 分岐 C（進捗率=100）
			15.修了証サービス → 修了証サービス：修了証発行(学生 ID, 受講登録 ID):修了証 ID
			16.修了証サービス → 通知サービス：通知送信(学生 ID, 件名, 本文):通知 ID
		# 条件分岐（修了）	* 分岐 D（進捗率<100） （この分岐では何も呼び出さない）
			図の右端へ出ていくアウトメッセージを 1 件追加する（送信元は通知サービス、宛先のライフラインは表示されない)。 17.通知サービス → ：外部通知配信(学生 ID, 件名, 本文):通知 ID
			図の左端から受講登録画面へ入ってくるインメッセージを 1 件追加する（宛先は受講登録画面、送信元のライフラインは表示されない)。 18. → 受講登録画面：通知(件名, 本文):void
			18 を一番下のメッセージとして表示すること。
		応答メッセージ	1.学生 → 受講登録画面：受講登録画面表示(学生 ID, セッション ID):メッセージ
			2.学生 → 受講登録画面：受講登録する(受講登録, セッション ID):メッセージ
			3.受講登録画面 → 受講登録サービス：受講登録作成(受講登録, セッション ID):受講登録 ID
			4.受講登録サービス → クーポンサービス：クーポン検証(クーポンコード, 学生 ID):検証結果
			5.受講登録サービス → 受講登録サービス：クーポン適用(受講登録 ID, クーポンコード):void
			6.受講登録サービス → 請求サービス：請求生成(受講登録 ID):請求 ID
			7.学生 → 受講登録画面：支払実行(請求 ID, 金額):メッセージ
			8.受講登録画面 → 支払サービス：支払登録(請求 ID, 金額):支払 ID
			9.支払サービス → 請求サービス：支払反映(請求 ID, 支払 ID):残額
			10.学生 → 受講登録画面：進捗更新(受講登録 ID, 進捗率):メッセージ
			11.受講登録画面 → 進捗サービス：進捗記録(受講登録 ID, 進捗率):進捗 ID
			12.学生 → 受講登録画面：修了証発行(受講登録 ID):修了証 PDF
			13.受講登録画面 → 修了証サービス：修了証発行(学生 ID, 受講登録 ID):修了証 URL
			14.修了証サービス → 進捗サービス：進捗取得(学生 ID, 受講登録 ID):進捗率
			15.修了証サービス → 修了証サービス：修了証発行(学生 ID, 受講登録 ID):修了証 ID
			16.修了証サービス → 通知サービス：通知送信(学生 ID, 件名, 本文):通知 ID
			図の右端へ出ていくアウトメッセージを 1 件追加する（送信元は通知サービス、宛先のライフラインは表示されない)。 17.通知サービス → ：外部通知配信(学生 ID, 件名, 本文):通知 ID
			図の左端から受講登録画面へ入ってくるインメッセージを 1 件追加する（宛先は受講登録画面、送信元のライフラインは表示されない)。 18. → 受講登録画面：通知(件名, 本文):void
		余計なメッセージ	余計なメッセージが存在しない
	ノート		ノート 1（メッセージ 6 の近く） クーポンが無効の場合でも請求は生成される。
			ノート 2（分割支払いループの近く） 支払は分割を許可するため、同一の請求 ID に対して複数回の支払登録が行われ得る。
SEQ_M+S（SEQ_M のチェックリストに追加）	配置場所（Layout）	9 つのライフラインを左から右に	学生、受講登録画面、受講登録サービス、クーポンサービス、請求サービス、支払サービス、進捗サービス、修了証サービス、通知サービス
			ライフラインの間隔は均等にし、メッセージ名が重ならない程度の横幅を確保する。
	色（Color / Paint）		受講登録画面は、ライフラインのヘッダ（名前表示部）を他と異なる背景色にする。
			請求サービスと支払サービスは、ライフラインのヘッダ（名前表示部）を同じ背景色にする。
			それ以外のライフラインのヘッダは、上記 3 種類の背景色とは異なる背景色にする。
			各ライフラインの活性区間には色を付ける。
			活性区間が重なっている場合は、色を変える。
	フォント（Typography）		ライフライン名の文字サイズは、メッセージ名の文字サイズより大きくする。
			太字にするのはライフライン名のみとし、メッセージ名とガード条件の文字は太字にしない。
	形状（Shape）		学生（actor）の表記は Awesome man（人物アイコン風）スタイルで表現する。

			ライフラインの活性区間を表示する。ネスト（呼び出しの入れ子）に合わせて重なって見えるようにする。
			ノートの形状は、六角形にする。
	可視性（Visibility）		ライフラインの下部のボックスは表示しない。

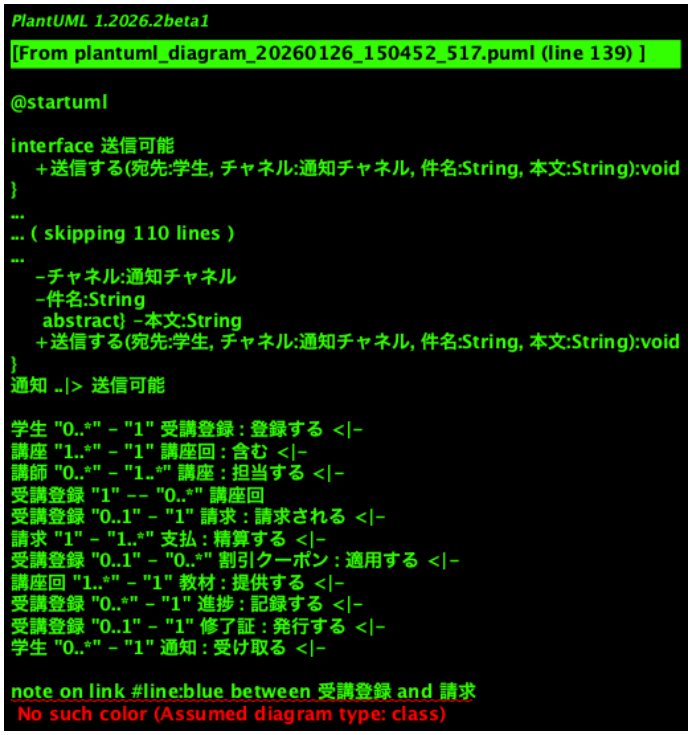
■ 付録 4 実行環境 E1（gpt-4.1）における実験結果

掲載形式は以下の通りである.

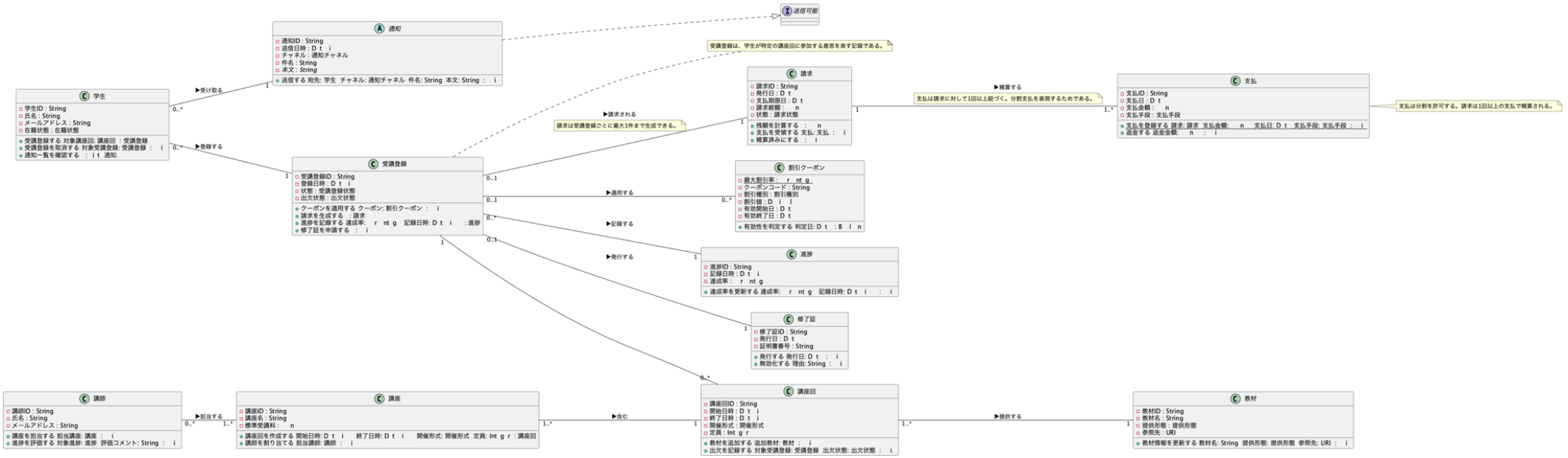
- タスク名（手法）

結果画像

- CLS_M（一回完結方式）



- CLS_M（ReAct-Dac）



- CLS-M+S（一回完結方式）

```
PlantUML 1.2026.2beta1
[From plantuml_diagram_20260127_075441_802.puml (line 15) ]

@startuml
skinparam classFontSize 18
skinparam classAttributeFontSize 14
skinparam classOperationFontSize 14
skinparam classFontStyle bold
skinparam classAttributeFontStyle normal
skinparam classOperationFontStyle normal


class 学生 as Student      (S   A    8  6)
 Syntax   rror? (Assumed diagram type: sequence)
```

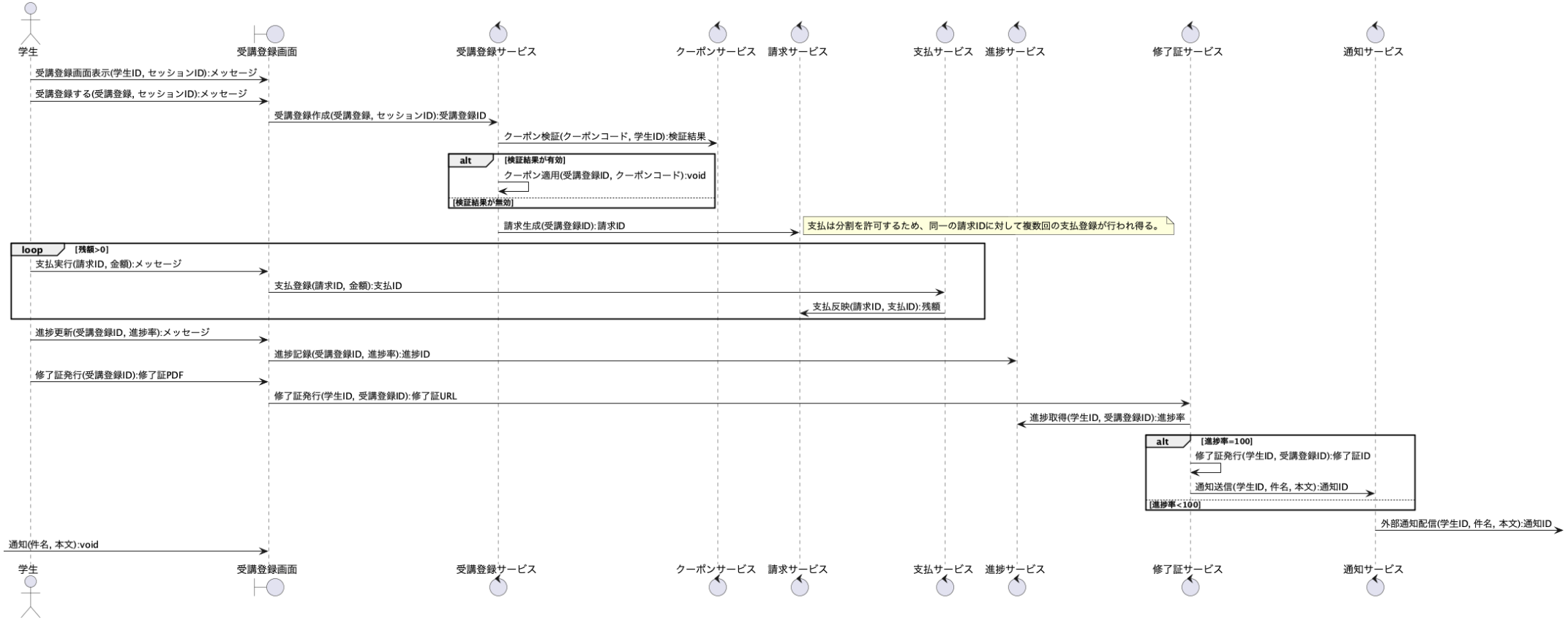
- CLS-M+S（ReAct-Dac）

```
PlantUML 1.2026.2beta1
[From plantuml_diagram_20260127_081630_331.puml (line 146) ]

@startuml
skinparam class {
   BackgroundColor #F9F9F9
   BorderColor Black
   FontSize 16
…
… ( skipping 115 lines )
…
   + 無効化する(理由 String)    oid
}
abstract class 通知 {
   - 通知ID   String
   - 送信日時   DateTime
   - チャネル   通知チャネル
   - 件名   String
   {abstract} - 本文   String
   + 送信する(宛先 学生, チャネル 通知チャネル, 件名 String, 本文 String)    oid
}
hide 通知 fields

interface 送信可能
通知 --|> 送信可能

学生 "0..*" -[#black]-> "1" 受講登録   登録する <|---|
講座 "1..*" -[#black]-> "1" 講座回   含む <|---|
講師 "0..*" -[#black]-> "1..*" 講座   担当する <|---|
受講登録 "1" -[#black]-> "0..*" 講座回
受講登録 "0..1" -[thickness=3,black]-> "1" 請求   請求される <|---|
 Syntax Error? (Assumed diagram type  class)
```

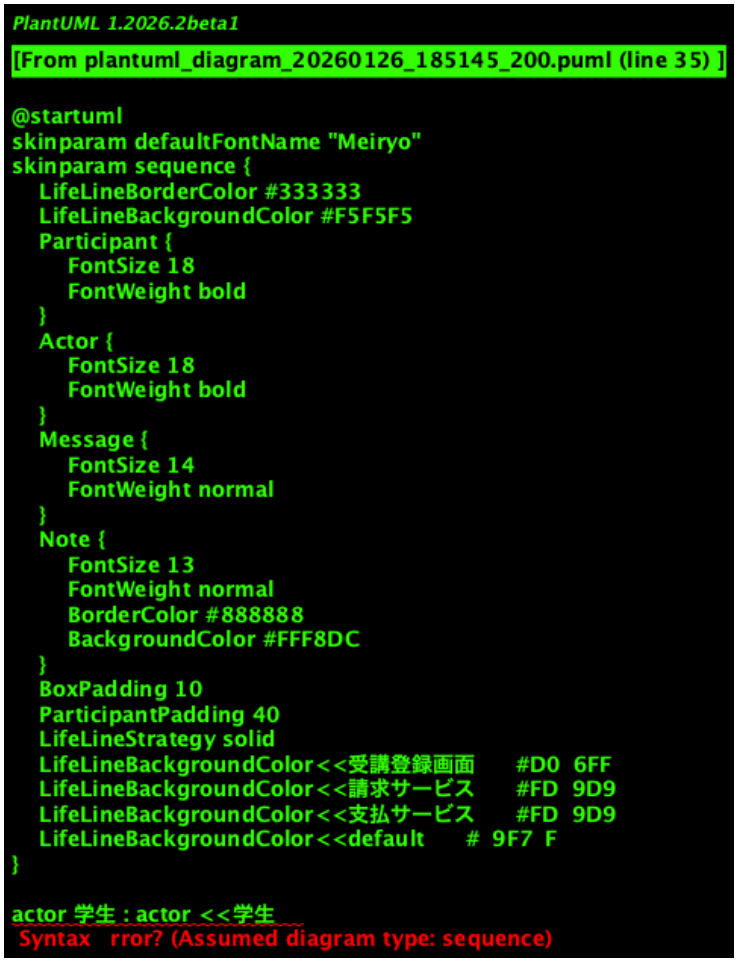
- SEQ-M（一回完結方式）


PlantUML 1.2026.2beta1
[From plantuml_diagram_20260126_122407_955.puml (line 47)]
@startuml
actor 学生 as tudent
boundary 受講登録画面 as egistration creen
control 受講登録サービス as egistration ervice
control クーポンサービス as Coupon ervice
...
... (skipping 20 lines)
...
tudent -> egistration creen : 支払実行(請求ID, 金額):メッセージ
egistration creen -> Payment ervice : 支払登録(請求ID, 金額):支払ID
Payment ervice -> Billing ervice : 支払反映(請求ID, 支払ID):残額
end
note right: 支払は分割を許可するため、同一の請求IDに対して複数回の支払登録が行われ得る。
tudent -> egistration creen : 進捗更新(受講登録ID, 進捗率):メッセージ
egistration creen -> Progress ervice : 進捗記録(受講登録ID, 進捗率):進捗ID
tudent -> egistration creen : 修了証発行(受講登録ID):修了証PDF
egistration creen -> Certificate ervice : 修了証発行(学生ID, 受講登録ID):修了証U L
Certificate ervice -> Progress ervice : 進捗取得(学生ID, 受講登録ID):進捗率
alt 進捗率=100
Certificate ervice -> Certificate ervice : 修了証発行(学生ID, 受講登録ID):修了証ID
Certificate ervice -> Notification ervice : 通知送信(学生ID, 件名, 本文):通知ID
else 進捗率<100
end
Notification ervice -> [外部] : 外部通知配信(学生ID, 件名, 本文):通知ID
yntax Error? (Assumed diagram type: sequence)

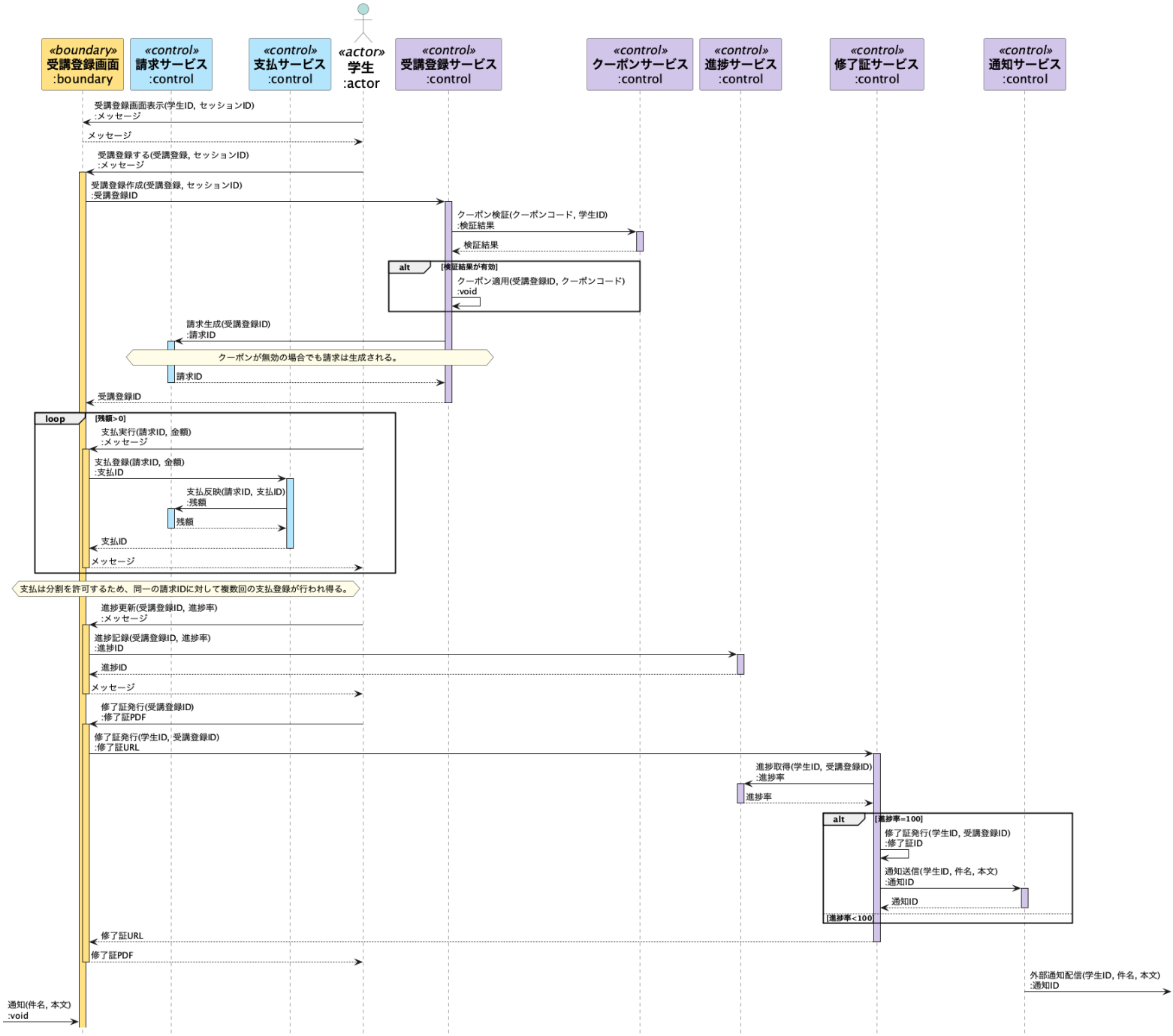
- SEQ-M（ReAct-Dac）



- SEQ-M+S（一回完結方式）



- SEQ-M+S（ReAct-Dac）



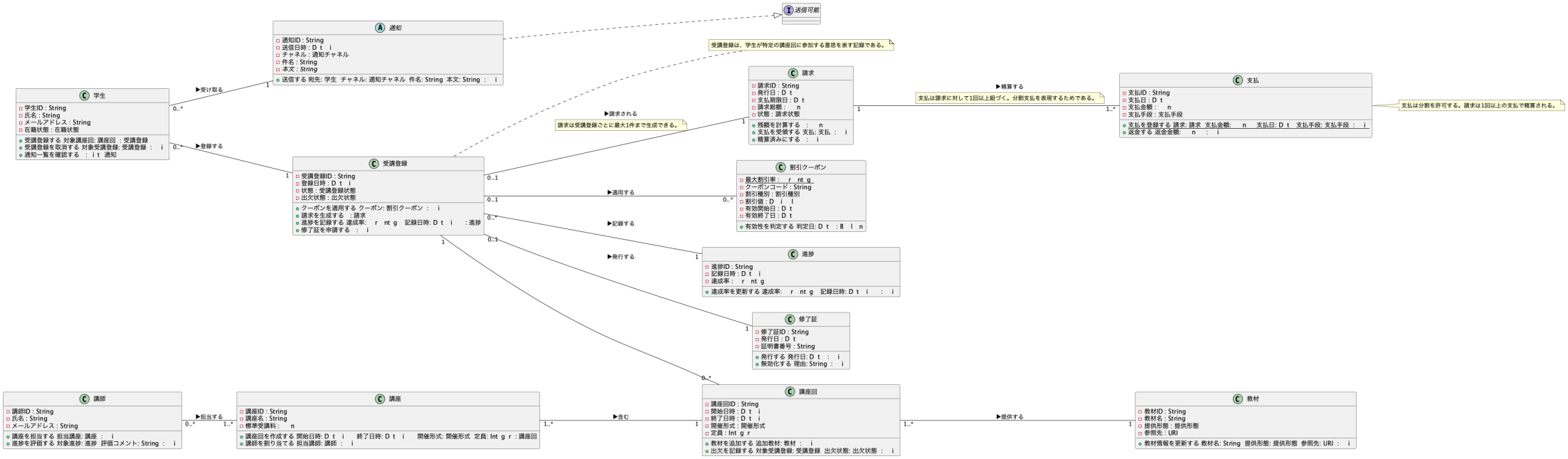
■ 付録 5 実行環境 E2（gpt-5.2）における実験結果

掲載形式は以下の通りである.

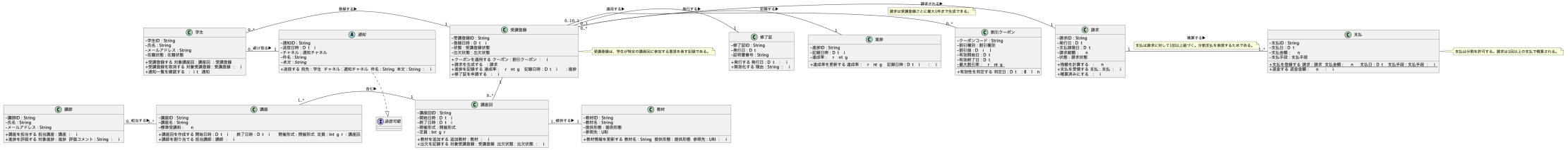
- タスク名（手法）

結果画像

- CLS-M（一回完結方式）

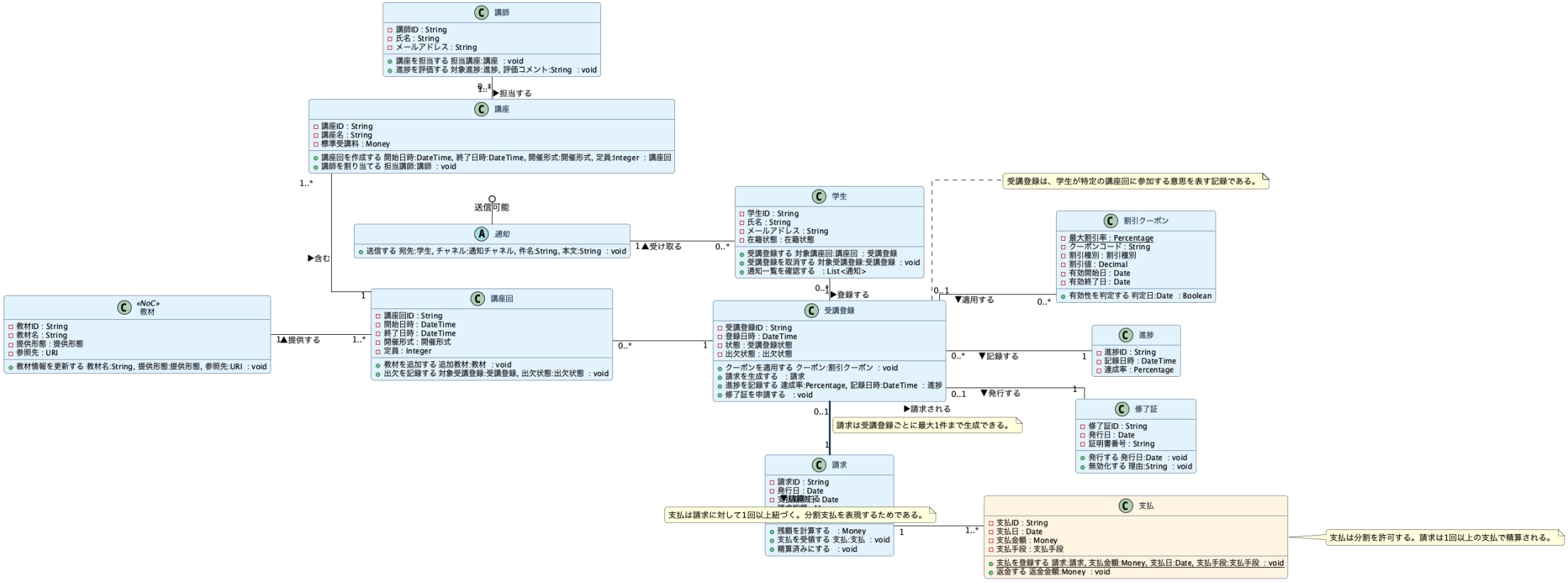


- CLS-M（ReAct-Dac）

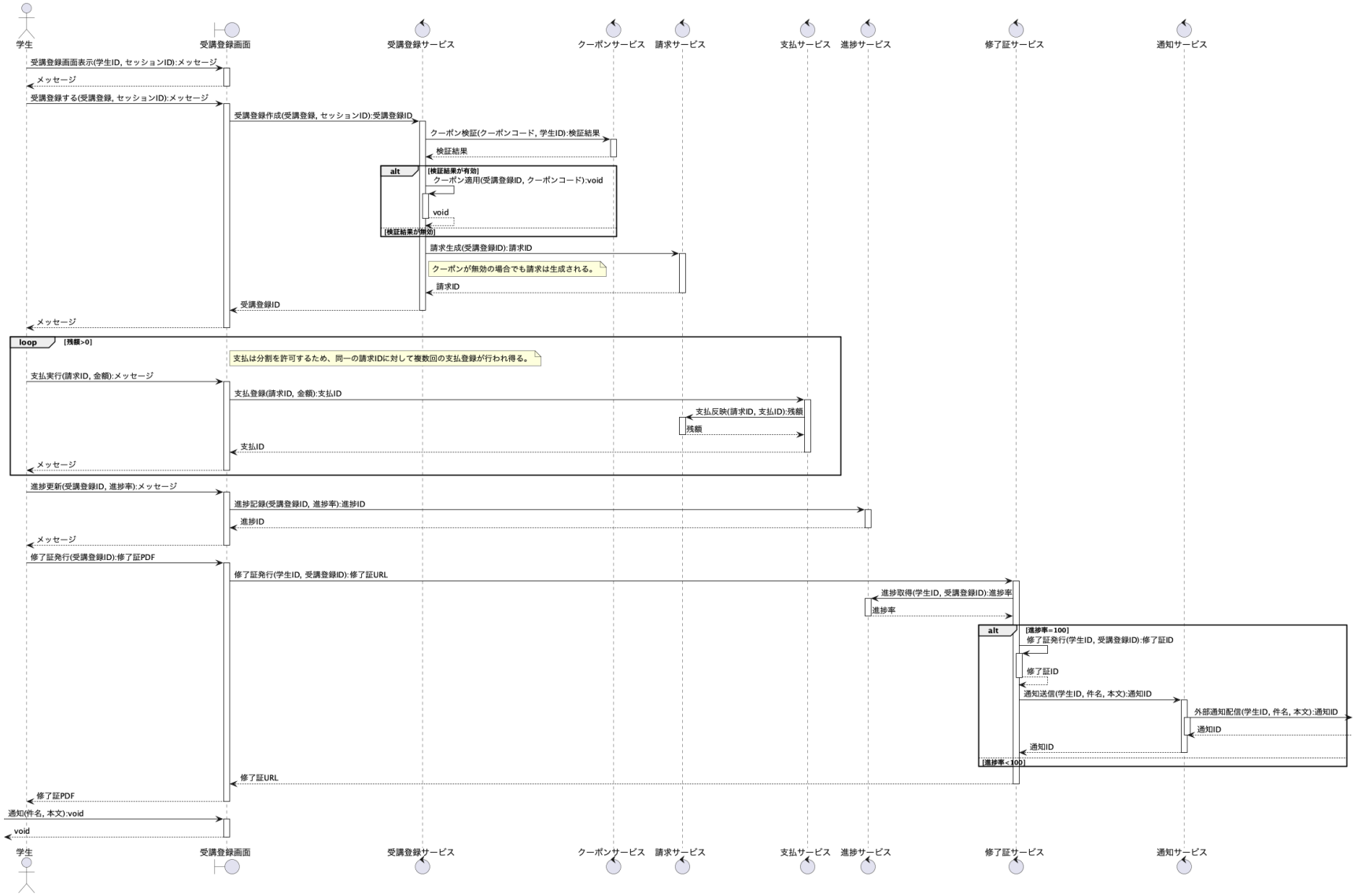


- CLS-M+S（一回完結方式）

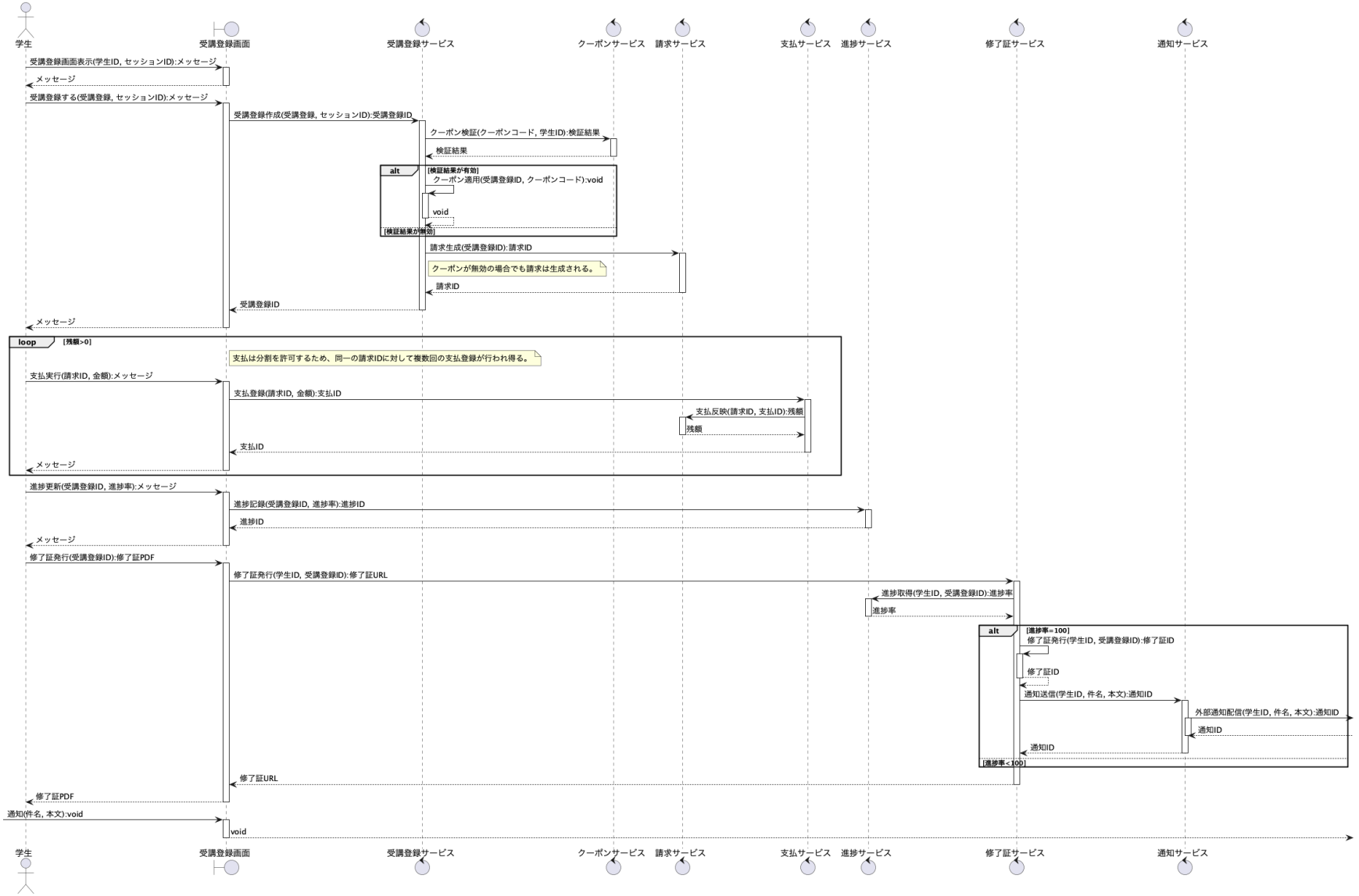
➢ CLS-M+S（ReAct-Dac）



- SEQ-M（一回完結方式）


学生
受講登録画面
受講登録サービス
クーポンサービス
請求サービス
支払サービス
進捗サービス
修了証サービス
通知サービス
受講登録画面表示(学生ID, セッションID):メッセージ
メッセージ
受講登録する(受講登録, セッションID):メッセージ
受講登録作成(受講登録, セッションID):受講登録ID
クーポン検証(クーポンコード, 学生ID):検証結果
検証結果
alt
[検証結果が有効]
クーポン適用(受講登録ID, クーポンコード):void
void
[検証結果が無効]
請求生成(受講登録ID):請求ID
クーポンが無効の場合でも請求は生成される。
請求ID
受講登録ID
メッセージ
loop
[残額>0]
支払は分割を許可するため、同一の請求IDに対して複数回の支払登録が行われ得る。
支払実行(請求ID, 金額):メッセージ
支払登録(請求ID, 金額):支払ID
支払反映(請求ID, 支払ID):残額
残額
支払ID
メッセージ
進捗更新(受講登録ID, 進捗率):メッセージ
進捗記録(受講登録ID, 進捗率):進捗ID
進捗ID
メッセージ
修了証発行(受講登録ID):修了証PDF
修了証発行(学生ID, 受講登録ID):修了証URL
進捗取得(学生ID, 受講登録ID):進捗率
進捗率
alt
[進捗率=100]
修了証発行(学生ID, 受講登録ID):修了証ID
修了証ID
通知送信(学生ID, 件名, 本文):通知ID
外部通知配信(学生ID, 件名, 本文):通知ID
通知ID
通知ID
[進捗率<100]
修了証URL
修了証PDF
通知(件名, 本文):void
void
学生
受講登録画面
受講登録サービス
クーポンサービス
請求サービス
支払サービス
進捗サービス
修了証サービス
通知サービス

- SEQ-M（ReAct-Dac）



- SEQ-M+S（一回完結方式）

- SEQ-M+S（ReAct-Dac）