Mamy:

1. Prosty wzorzec (przykład) użycia operatorów w łańcuchu akcji “autonomicznej” pętli (autonomiczna na razie z nazwy, nie zawiera funkcji “learning”). W tych ramach
   * przykład definiowania pętli z użyciem “master CR” I “master operator”
   * przykładowy mechanizm wyzwalania kolejnych bloków funkcjonalnych (operatorów) poprzez modyfikowanie odpowiednich CR (niedoskonały: znacznik czasowy do detekowania “amiany” nawet jeśli zmiany nie ma; być może tak musi zostać, ale można pomyśleć nad sposobem sygnalizowaniam, które konkretne atrybuty CR uległy zmianie, aby operator nie robił wszystkiego “od początku”)
   * przykład zdefiniowanie (użycia) “policy”
2. Wiedzę na temat użycia rego (na razie “czyste” rego: zapytania HTTP + serwer)
3. Czego nie ma / otwarte kwestie ponad to, co wskazano powyżej:
   * wbudowanie odwołań do rego w CR i w operatorze
     + definicja w master CR, wykorzystanie przez operator na bazie definicji we właściwym CR
     + role w kontekście rego – możliwa propozycja:
       - *policy* – specyfikacja (“program”) w języku rego
       - blok *input* w wywołaniu GET zawiera dwie sekcje: *policy-params* oraz *check-data*
         * “policy-params” to dynamiczna paramatryzacja wywołania policy w konkretnej instancji pętli/konkretnym wywołaniu – podawane we właściwym CR
       - sekcja */v1/data* – baza danych / dodatkowe parametry “statyczne” dla policy, ładowana off-line (choć teoretycznie parametry te mogą też być uzupełniane online/podczas działania pętli)
   * dobra konwencja nazewnictwa, aby w automatyczny sposób wywodzić jednoznacznie identyfikowalne nazwy CR / operatorów, ścieżki w rego odpowiadające konkretnej instancji pętli, etc.)
   * sposób wywoływania/korzystania z/realizacji bardziej skomplikowanych obliczeń. Prosty przykład zaczerpnięty pętli deliberatywnej naszego demo: pętla ta wywodzi decyzję na podstawie odpowiednio przetwarzanej historii ostatnich 10 akcji na każdym z zasobów (cpu, memory) i trudno takie “obliczenie” wykonać w rego, natomiast sam wynik obliczenia powinien być w rego możliwy do wykorzystania. Sugestia rozwiązania: zaprojektować łańcuch (cykliczny) wywołań: rego –> algorytm-wywoływany-w operatorze działający z użyciem dokumentu *result* z rego -> rego z użyciem wyniku działania algorytmu podawanego w formie dokumentu *input* …
   * w rezultatach (*result*) otrzymanych z *policy* może znajdować się wskazanie wywołania konkretnego “aktora”. Aktorem może być konkretny blok pętli (operator); na razie jest to statycznie zapisane w CR (operator najwyżej dokleja prefix do nazwy i w ten sposób zna odpowiedni CR - instancję). Jak to dobrze zrobić, także trzymając się dobrej konwencji nazewnictwa dwie kropki wyżej? Aktorem może też być zewnęrzna funkcja, np. jakaś aplikacja ML, którą chcemy o coś zapytać. Ten ostatni przypadek troche wiąże się też z poprzednim punktem - "kropką”.
   * Pytanie filozoficzne: czy możliwy jest “generyczny” typ operatora z całą logiką zawartą w odpowiednio skonfigurowanych *policy*, *data*, *input* oraz aplikacjach zewnętrznych? Czyli taki “data driven” kręciołek?