|  |  |  |
| --- | --- | --- |
| Nazwa | Zalety | Wady |
| SOAP | 1. niezwykłą elastyczność protokołu, który pozwala przenosić właściwie dowolne informacje 2. możliwość definiowania struktury i semantyki przenoszonych informacji 3. możliwość łączenia z różnymi protokołami transportowymi (np. HTTP) 4. możliwość realizacji różnych scenariuszy komunikacji 5. akceptowalność protokołu przez właściwie wszystkie systemy komputerowe i środowiska systemowe 6. niezawodność protokołu dzięki ścisłemu zdefiniowaniu sytuacji wystąpienia błędu oraz zachowania aplikacji w takich okolicznościach | 1. duży narzut samego języka XML (rozmiar komunikatu jest znacząco większy niż sumaryczny rozmiar danych w nim zawartych) 2. jest jeszcze dość młodym protokołem, podlega rozwojowi i modyfikacjom (chociaż jest już dość dobrze i ściśle zdefiniowany) 3. trudność w utrzymaniu aplikacji klienta |
| RMI | 1. Prostota jego uzytkowania, w porownaniu np z COBRA, RMI jest proste i malo skomplikowane 2. Troszczy sie o szczegoly przesylania obiektow ich serializacje i deserializacje 3. Zapewnia niezaleznosc programow od procesora wystarczy dzialajaca maszna Javy | 1. wsparcie tylko dla programow napisanych w javie 2. zarowno klient jak i serwer musza byc aplikacjami lub apletami javy i nie mozna komunikowac sie z programami napisanymi w zadnym innym jezyku programowaniu za pomoca tego mechanizmu   (o ile użycie dla klienta Javy jesty ok o tyle dla serwera jest często nie możliwe). |
| Sockety | 1. Wydajne, 2. Niski narzut na ruch sieciowy, 3. Wysłanie tylko zaktualizowanej informacji. | 1. Problemy z bezpieczeństwem, 2. Klient i serwer musza posiadać mechanizmy pozwalające zinterpretować dane. |
| REST | 1. Prostota, 2. Mniejszy narzut obliczeniowy, 3. Wykorzystanie znanej i przetestowanej infrastruktury: Web, 4. Możliwość stosowania serwerów pośredniczących, 5. Mniej problemów ze współoperacyjnością (jednolity interfejs) 6. Minimum narzędzi potrzebnych do implementacji | 1. Dodaje znikome opóźnienia, 2. Żądania nie są wystarczające dla długich ciągów danych, |
| CORBA | 1. Architektura CORBA jest otwartym rozwiązaniem opartym na opublikowanej specyfikacji 2. Jest niezależna od sprzętu i systemu operacyjnego. Współdziałające komponenty mogą działać na różnych architekturach sprzętowych i pod kontrolą różnych systemów operacyjnych. 3. Obiekt programowy zgodny z architekturą COBRA posiada ściśle zdefiniowany interfejs, poprzez który odbywa się komunikacja. Zmiany w implementacji obiektu nie mają wpływu na inne obiekty, o ile zostanie zmieniony interfejs. 4. Komunikacja pomiędzy obiektami programowymi zgodnymi z COBRA odbywa się przy wykorzystaniu IIOP. Obiekty programowe mogą ze sobą w pełni współpracować, nawet jeżeli działają na różnych systemach operacyjnych i zostały utworzone z wykorzystaniem różnych języków programowania. 5. Obiekty zbudowane na jednej platformie mogą być wykorzystane z każdej innej z obsługiwanych platform 6. Budowa aplikacji odbywa się zgodnie z zasadami techniki obiektowej. 7. Dostęp do obiektów bez konieczności określania ich położenia. | 1. Brak standardowego i szeroko zaimplementowanego mechanizmu bezpieczeństwa 2. Przywiązanie do szczegółów technicznych języków niskiego rzędu 3. Trudność we współdziałaniu i przenośności 4. Konieczna jest komunikacja makroskopowa 5. Problemy z bezpieczeństwem. |
| MPI | 1. Wysoka wydajność, 2. Efektywna obsługa dużej liczby procesów 3. Dobra dokumentacja 4. Bogata biblioteka funkcji | 1. Złożony sposób tworzenia programów równoległych, 2. Statyczna konfiguracja jednostek przetwarzających, |