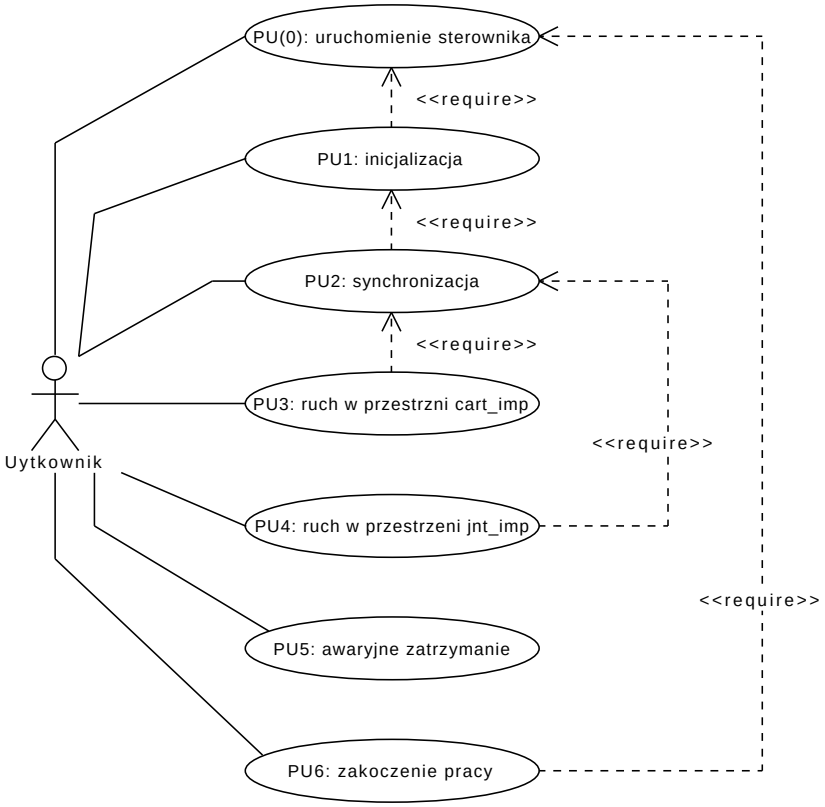
Specyfikacja sterownika robota Velma

Jakub Postępski

7 stycznia 2019

Robot Velma (rys. 1) został skonstruowany w laboratorium robotyki. Posiada dwa ramiona sterowane impedancyjnie i może obracać tułowiem. Robot może działać w swojej przestrzeni konfiguracyjnej i operacyjnej.

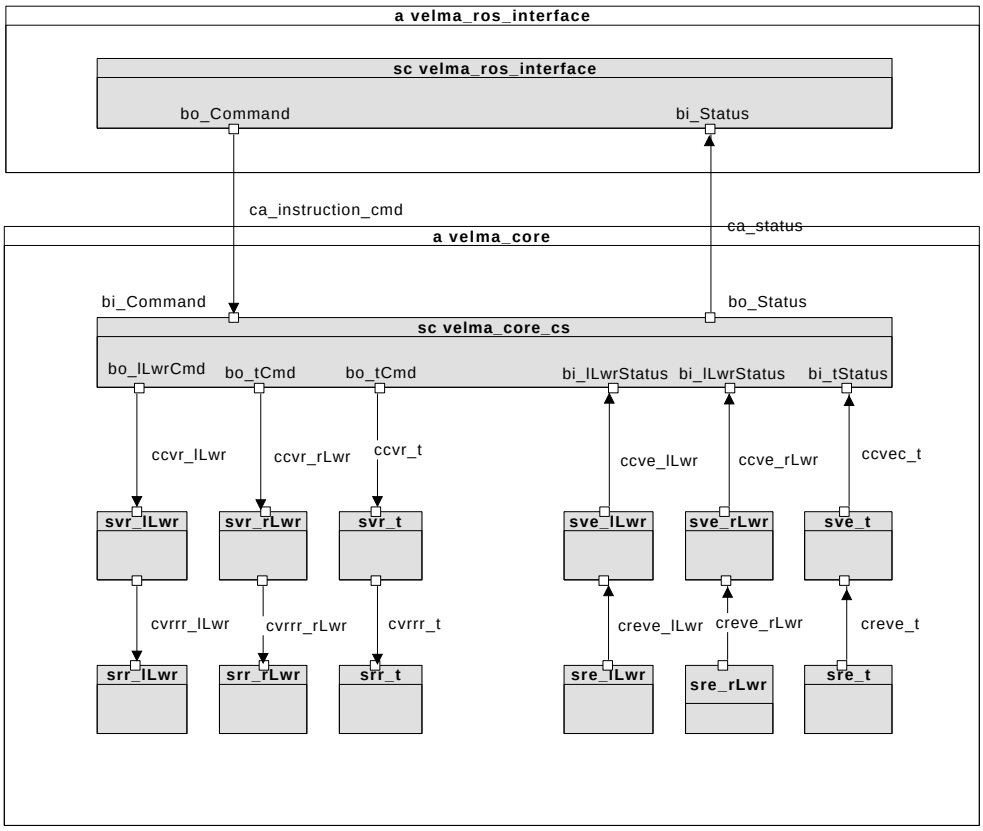


Rysunek 1: Przypadki użycia systemu

1 Struktura agentów systemu

Sterownik robota został zdefiniowany jako dwa agenty (rys. 2). Agent **velma** steruje pracą robota i kontroluje jego urządzenia. Agent **velma_ros_interface** służy do wydawania poleceń dla robota i kontroli stanu.

W sprawozdaniu szczegółowo opisany jest tylko agent **velma**.



Rysunek 2: Struktura agentów

Poniżej przedstawiono zawartość kanałów informacyjnych robota.

ca_instruction_cmd	Opis	Typ
inst	instrukcje sterowania	struktura
ca_instruction_st	Opis	Typ
stat	status robota	struktura

Tabela 1: Zawartość kanałów komunikacyjnych CA

ccve_lLwrCmd	Opis	Typ
lt	zadane momenty stawów lewego ramienia	float[7]
ccve_rLwrCmd	Opis	Typ
rt	zadane momenty stawów prawego ramienia	float[7]
ccve_tCmd	Opis	Typ
tq	zadany obrót tułowia	float
th	polecenie zaczęcia synchronizacji obrotu tułowia	bool
te	polecenie załączenia silnika tułowia	float

Tabela 2: Zawartość kanałów komunikacyjnych CCVE

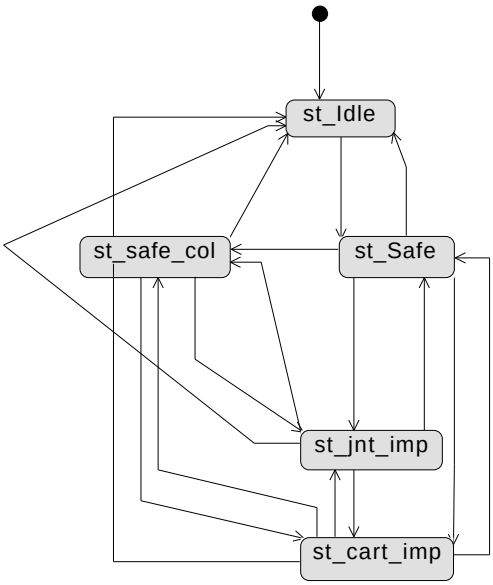
cvec_lLwrStatus	Opis	Typ
lt	pozycja stawów lewego ramienia	float[7]
ls	informacje diagnostyczne lewego ramienia	bool
cvec_rLwrStatus	Opis	Typ
rs	informacje diagnostyczne prawego ramienia	bool
rt	pozycja stawów prawego ramienia	float[7]
cvec_t	Opis	Typ
tsh	synchronizacja stawu tułowia w toku	bool
tse	silnik stawu tułowia aktywny	bool
tsq	pozycja stawu tułowia	float

Tabela 3: Zawartość kanałów komunikacyjnych CVEC

2 Specyfikacja podystemu sterowania

Podsystem sterowania robota (rys. 3) ma za zadanie wykonywać zadania narzucone przez agenta **velma_ros_interface**. Jest odpowiedzialny za generację trajektorii oraz wyliczanie praw sterowania. Kontroluje podsystem wirtualnego efektora i zapobiega sytuacjom awaryjnym. Inicjalizuje włączanie i synchronizację silników.

Podstem działa w oparciu o maszynę stanów (rys. 3). Stany są przełączane na podstawie predykatów (tab. 4).



Rysunek 3: Automat skończony podsystemu sterowania

Nazwa predykatu	Opis
CURRENT_BEHAVIOR_OK	podsystem sterowania pracuje normalnie
recvOneCmd	otrzymano dokładnie jedną wiadomość sterowania
recvCartImpCmd	otrzymano wiadomość sterowania w trybie cart_imp
recvJntImpCmd	otrzymano wiadomość sterowania w trybie jnt_imp
recvSafeColCmd	otrzymano wiadomośc pracy w trybie safe_col
inSelfCollision	robot jest w stanie autokolizji
IN_ERROR	wykryto błąd systemu

Tabela 4: Definicja predykatów

Do każdego stanu przyporządkowano zachowania (tab. 5, tab. 6). Zachowanie **idle** jest osiągane po włączeniu robota. Zachowanie **safeCol** pozwala poruszać się w trybie unikania kolizji. Zachowanie **cartImp** odpowiada sterowaniu w przestrzeni operacyjnej a zachowanie **jntImp** w przestrzeni konfiguracyjnej. Zachowanie **safe** jest osiągane przy wykryciu błędu systemu.

Stan	Zachowanie
st_idle	idle
st_safe	safe
st_cart_imp	cart_imp
st_jnt_imp	jnt_imp
st_safe_cold	safe_col

Tabela 5: Przyporządkowanie zachowań do kolejnych stanów podsystemu sterowania

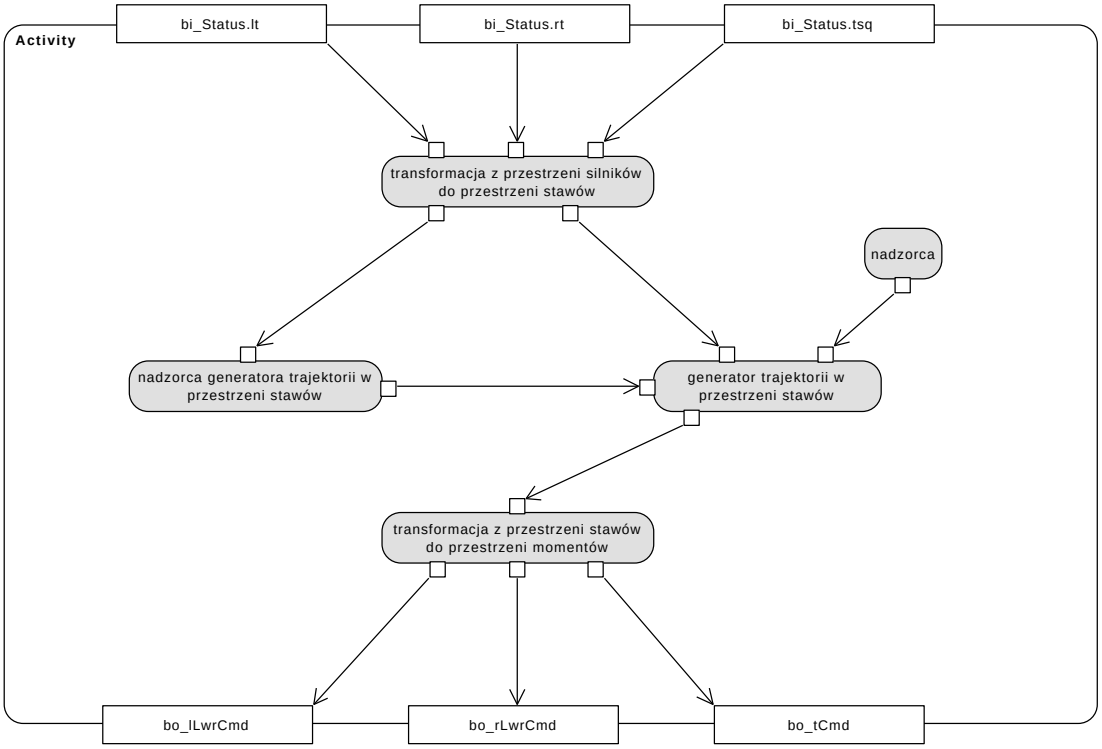
Stan początkowy	Stan końcowy	Warunki predykatów
st_safe	st_cart_imp	$\neg IN\_ERROR \wedge recvCartImpCmd$
st_safe	st_jnt_imp	$\neg IN\_ERROR \wedge recvJntImpCmd$
st_safe	st_safe_col	$\neg IN\_ERROR \wedge recvSafeColCmd$
st_safe_col	st_cart_imp	$\neg IN\_ERROR \wedge recvCartImpCmd$
st_safe_col	st_jnt_imp	$\neg IN\_ERROR \wedge recvJntImpCmd$
st_cart_imp	st_safe	$\neg IN\_ERROR \wedge \neg recvOneCmd$
st_cart_imp	st_jnt_imp	$\neg IN\_ERROR \wedge \neg recvOneCmd \wedge recvJntImpCmd$
st_cart_imp	st_safe_col	$\neg IN\_ERROR \wedge \neg recvOneCmd \wedge recvSafeColCmd$
st_jnt_imp	st_safe	$\neg IN\_ERROR \wedge \neg recvOneCmd$
st_jnt_imp	st_cart_imp	$\neg IN\_ERROR \wedge \neg recvOneCmd \wedge recvCartImpCmd$
st_jnt_imp	st_safe_col	$\neg IN\_ERROR \wedge \neg recvOneCmd \wedge recvSafeColCmd$
dowolny	idle	IN_ERROR

Tabela 6: Warunki przejść pomiędzy stanami podsytemu sterowania

Przykładowa funkcja przejścia **jntMove** (rys. 4, rys. 5) jest odpowiedzialna za wykonanie ruchu w przestrzeni konfiguracyjnej.

Zachowanie	Funkcja przejścia	lt	rt	tq	th	te	ls	rs	tsq	tse	tsh
idle	passive	x	x								
safe	enableAndHome				x	x				x	x
	passive	x	x	x							
	errorHandling						x	x	x		
jntImp	jntMove	x	x	x					x		
	errorHandling						x	x	x		
cartImp	cartMove	x	x	x					x		
	errorHandling				x	x	x	x	x		
safeCol	safeColMove	x	x	x							
	errorHandling				x	x	x	x	x		

Rysunek 4: Kompozycja funkcji przejść



Rysunek 5: *Fp* jntMove