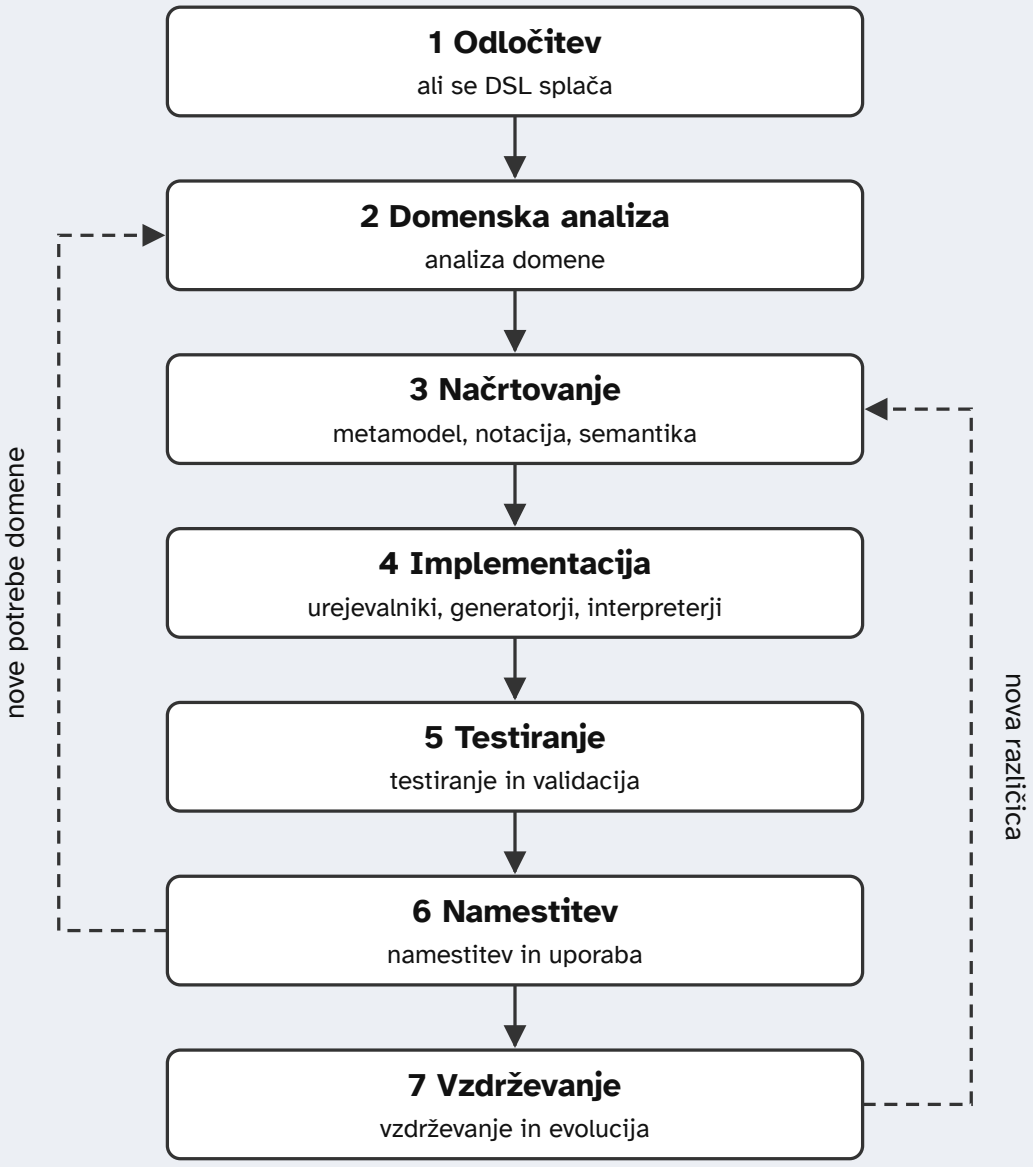
Življenjski cikel DSL

1 Kaj je DSL

Domensko specifični modelirni jezik (DSL) je jezik, zasnovan za eno določeno domeno (npr. avtomatizacija stavb, poslovni procesi, državni avtomati). Za razliko od splošnih jezikov (UML, Java) vsebuje samo pojme, ki jih domena potrebuje.

2 Življenjski cikel



Slika 1: Življenjski cikel DSL

Cikel ni linearen. Povratne puščice pokažejo, da se jezik nenehno popravlja glede na uporabo in spremembe domene.

3 Koraki

3.1 1. Odločitev

Kaj naredi: Presodi, ali je DSL sploh smiselen. Primerja stroške razvoja in vzdrževanja s koristjo (hitrejše modeliranje, manj napak, boljša komunikacija z domenskimi strokovnjaki) in preveri, ali obstoječi jezik ne zadostuje.

Rezultat: odločitev za ali proti DSL, utemeljitev, obseg projekta.

Pozor: Če domena še ni stabilna ali je primerov premalo, je splošni jezik ali knjižnica pogosto boljša izbira.

3.2 2. Domenska analiza

Kaj naredi: Ugotovi, kateri pojmi, odnosi in pravila obstajajo v domeni. Sodelujemo z domenskimi strokovnjaki in zbiramo primere uporabe.

Rezultat: glosar pojmov, seznam primerov uporabe, meje domene.

Pozor: Če je domena preozka ali preširoka, bo jezik neuporaben. Omejitve zapiši že tukaj.

3.3 3. Načrtovanje

Kaj naredi: Oblikuje jezik. Določi abstraktno sintakso (metamodel: pojmi, lastnosti, povezave, omejitve), konkretno sintakso (kako model izgleda: besedilo, grafika, tabele) in semantiko (pravila preverjanja in pomen modelov).

Rezultat: metamodel, opis notacije, pravila semantike.

Pozor: Metamodel naj bo minimalen. Notacija mora biti razumljiva domenskim strokovnjakom, ne samo razvijalcem.

3.4 4. Implementacija

Kaj naredi: Izdela orodja: urejevalnik, preverjalnik, generator kode ali interpreter. Pogosto se uporabi jezikovno ogrodje (npr. Xtext, MPS, Langium).

Rezultat: delujoče orodje za urejanje in obdelavo modelov.

Pozor: Orodja naj se gradijo iz metamodela, ne ročno, da ostanejo skladna.

3.5 5. Testiranje

Kaj naredi: Preveri, ali jezik pokrije primere uporabe iz domenske analize. Napiše testne modele in preveri, ali so napake res zaznane.

Rezultat: testni modeli, poročilo o pokritosti, seznam napak.

Pozor: Najbolje je testirati s pravimi domenskimi primeri, ne le s sintetičnimi.

3.6 6. Namestitev

Kaj naredi: Jezik in orodja dobijo uporabniki. Pišejo modele, generirajo rezultate in jih uporabljajo v praksi.

Rezultat: nameščena orodja, dokumentacija, primeri, dejanski modeli in artefakti.

Pozor: Brez dokumentacije in primerov uporabniki jezik obidejo ali ga napačno uporabljajo.

3.7 7. Vzdrževanje

Kaj naredi: Zbira povratne informacije uporabnikov, odpravlja napake in uvaja nove različice jezika. Upravlja združljivost s starimi modeli (migracije, verzije metamodela) in posodablja orodja.

Rezultat: nova verzija jezika, migracijske skripte, zapiski o spremembah.

Pozor: Spremembe metamodela lahko pokvarijo obstoječe modele. Verzioniranje je obvezno.

4 Povzetek

Korak	Ključni vhod	Ključni izhod
1 Odločitev	potreba, stroški	odločitev, obseg
2 Domenska analiza	domenski eksperti	glosar, primeri
3 Načrtovanje	glosar, primeri	metamodel, notacija, semantika
4 Implementacija	metamodel, semantika	orodja
5 Testiranje	orodja, primeri	testi, napake
6 Namestitev	orodja, dokumentacija	modeli, artefakti
7 Vzdrževanje	povratne informacije	nova različica