značilno, da so bile ročne. Posamezne faze dela so se zaključile same vase, tako da je celotni cikel spominjal na kaskadni slap. Zahteval je veliko ročnega dela, zato je zlahka prišlo do napak. Vsaka faza je zahtevala svoj jezik. Pri opredelitvi sistemskih specifikacij je bilo premalo formalnosti. Uporabljena orodja so bila zahtevna, vendar nekonsistentna. Verifikacija in vrednotenje sistema je bilo možno šele na koncu cikla (v fazi testiranja).

Po l. 1985 so nastale metodologije, ki uvajajo avtomatska razvojna orodja, se izogibajo ročnemu kodiranju, kjer je to smiselno, vpeljujejo avtomatsko zasnovo in programiranje. Pravilnost zasnove preverjajo računalniško.Današnja gradnja kvalitetnih sistemov zahteva tolikšno natančnost, da je ročne metode ne morejo ponuditi. Zato je razumljiva zahteva po izboru kvalitetne in sodobne metodologije, ki je podprta z avtomatskimi razvojnimi orodji.

Pojavilo se je več vrst življenjskih ciklov, ki bolje zadostijo potrebam kot klasični cikel:

- Življenjski cikel z enojno iteracijo. Sem sodi tudi klasični cikel, vendar danes uporabljajo orodja za izdelavo prototipov, za avtomatsko zasnovo, generatorje kod, kar je spremenilo njegovo naravo.
- Evolucijski življenjski cikel. V tem ciklu nastaja zaporedje prototipov, ki se bližajo ciljnemu sistemu. Končni prototip lahko preraste v delujoči sistem.
- Življenjski cikel časovnih omejitev. To je zelo uspešni način kako vključiti izdelavo prototipov, tehniko SZA, avtomatsko zasnovo in generacijo kode. Tako je pospešen razvoj, stroški in riziki razvoja pa so minimizirani.
- Življenjski cikel hitrih rezultatov. Uporabljan je pri razvoju enostavnih aplikacij. Ne potrebuje niti formalnih specifikacij niti zasnove. Uporabljajo ga zlasti v IC.
- Življenjski cikel za razvoj sistemov za podporo odločanju. Sistem za podporo odločanju pomaga pri boljših poslovnih odločitvah na določenem področju. Razvijajo jih praviloma uporabniki na inovativen način, brez specifikacij.