Zapuščina napredne informatizacije

1. Predgovor

Pričujoči zbornik prispevkov s simpozija o Zapuščini napredne informatizacije, ki je potekal v [Računalniškem muzeju](https://www.racunalniski-muzej.si/) v Ljubljani 13. junija 2022, obravnava raznolike poglede na izzive in potrebe po sistematični hrambi programske računalniške dediščine. Obenem pa samo problematiko uvaja v sodoben slovenski strokovni diskurz z namenom, da se postavi osnova za razmislek o prihodnjih ustrojih, ki bi nam kot družbi pomagali, da zagotovimo dolgoživost spomina na digitalni sloj človeškega udejanjanja.

V Računalniškem muzeju smo potrebo za ta skupni razmislek občutili kot pereč problem v lastnih dejavnostih medijske in digitalne arheologije ter v procesih in poskusih rekonstrukcije prelomnih digitalnih odtisov slovenske družbe. Simpozij smo sklicali s tezo, da prepogosto prepuščamo pozabi tisto, s čimer imamo opravka vsak dan, ko kot uporabniki digitalne tehnologije ignorantsko zanemarjamo plast, ki nam omogoča to nič več čudežno, a vendar vedno manj razumljeno vmesniško plast med človekom in strojem. Kar se je ohranilo, se je ohranilo tako rekoč slučajno. Zahvala gre zavednim posameznicam in posameznikom, ki so delovali bodisi znotraj organizacij bodisi v zasebni kapaciteti, zato želimo z izdajo zbornika postaviti ilustrativne temelje za nadaljnjo obravnavo in usmeritev v iskanje sistematičnega nagovora k ohranjanju tistega, kar iz nežive strojne opreme ustvari digitalno človeško izkušnjo.

Nabor strokovnjakov, ki so sodelovali na posvetu, je bil namenoma raznolik, zato da smo zajeli širino pogledov in razmislekov od prvoosebnih izrazov ustvarjalcev. Ti so s svojim delovanjem prispevali k naboru izjemno pomembnih prelomnih trenutkov softverskega razvoja v Ljubljani in Sloveniji, vključno s humanističnimi interpretacijami in dobrimi praksami, ki so morda že vzpostavljene v drugih segmentih raziskovalnega dela ali v posameznih visoko reguliranih industrijah. Odprli smo pravna vprašanja, povezana z nejasno opredelitvijo programske intelektualne lastnine, ter izpostavili učinke, ki jih je povezano neprekinjeno delovanje na tem področju prineslo razvoju mesta in nenazadnje samozavesti današnjih razvijalcev programskih rešitev, ki so se kalili v okolju z bogato razvejano institucionalno in samoniklo softversko dejavnostjo.

V nagovorni predstavitvi dr. Roberto di Cosmo uokviri namen posveta s predstavitvijo geneze in delovanja fundacije UNESCO Software Heritage, katere ambasadorji smo tudi v Računalniškem muzeju. Poda izhodiščne zagovore za oblikovanje statusa softverske nesnovne dediščine, kot ga vidi UNESCO, in demonstrira določen del na praktičnem prikazu iz hrambe SWH. Praktične prikaze in pregled večdesetletnega delovanja na področju razvoja softverskih rešitev prispevajo dr. Saša Divjak s področja računalniški ved, dr. Primož Jakopin s področja jezikoslovja in Franc Zakrajšek s področja urbanističnega načrtovanja in digitalizacije arhitekturne kulturne dediščine. Divjak se osredotoča na preglavice, ki jih povzroča sprotno zastaranje programskih rešitev, Jakopin pa odpre vprašanja arhiviranja z vidika poznejše muzejske predstavitve. Poglobljen pregled razvoja digitalizacije specifičnega področja – urbanizma – nam predstavi Zakrajšek, ki skozi tehnološke rešitve jasno prikaže, kako je razvoj v Ljubljani držal korak s časom. Nadalje mag. Silvana Žorž poveže teoretske in praktične iztočnice v prispevku o vrednotah v softverskem razvoju, kot sledijo iz analize poslovne in programerske kulture podjetja HERMES SoftLab, in odpre vprašanje zmožnosti ohranjanja tako neotipljive, a vendar bistvene zapuščine, kot je vrednostni sistem. To človeško noto nagovori tudi Boštjan Špetič v prispevku o rekonstrukciji računalnika Iskra Delta Partner, arheoloških najdbah s trdega diska in dejanskem učinku prezentacije iz muzejske prakse. Dr. Andrej Pančur osvetljuje dobre prakse in regulativne okvire na področju digitalne humanistike. Dr. Nataša Milić-Frayling pa ilustrira dobre prakse na področju dolgotrajne hrambe softvera skozi primer iz visoko reguliranega industrijskega okolja ter zahtevnega umetniškega okolja, kjer je istovetnost uporabniške izkušnje na prvem mestu. Matija Šuklje pa nam oriše kompleksnost pravnih interpretacij programja, ki se jih moramo zavedati ob razmisleku o sistematični hrambi, rekonstrukciji in predstavljanju javnosti.

Posvet je vprašanja odpiral, a obenem ponudil nastavke za oblikovanje zaključkov. Vodilna misel, ki se je prepletala skozi vsa predstavljena stališča, pa je bila izjemno človeška.

Softver je minljiv.

Softver je minljiv, a obenem nesmrten v svoji ideji, udejanjanju človeške iznajdljivosti, enkapsuliranju duha določenega časa in vrednostnih sistemov določene skupine ljudi. Je izraz naše kulture. Je naša dediščina.

V Sloveniji je prostora za izboljšavo na področju hrambe, evidentiranja, konzervacije, restavriranja softverske dediščine in nenazadnje interpretacij in prezentacij za javnost še ogromno. Prvi korak pa je zagotovo narejen s tem, ko se zavemo, da gre tudi pri programju za nesnovno kulturno dediščino, vredno celostne obravnave ali pa za vedno zapisano pozabi. Zgodovina, o kateri razpravljajo avtorji, je resnično nedavna, stara komaj več kot pol stoletja. Izgublja pa se s svetlobno hitrostjo. Z vsako nadgradnjo uporabniškega sistema, z vsako novo generacijo strojne opreme. Zato je zdaj pravi čas, da se vprašamo, kaj bomo lahko pokazali zanamcem čez naslednjih 50 let. Jim bomo brez istovetnih interaktivnih prikazov res lahko dovolj dobro razložili, kako so nas kot družbo oblikovale digitalne interakcije? Bomo kolektivno pozabili nanje?

Gaja Zornada,

vodja Računalniškega muzeja, Ljubljana, julij 2022

1. 1. Dediščina programske opreme: svetovni arhiv izvornih kod programske opreme

Roberto Di Cosmo

Programska oprema ni le gonilo naše panoge in inovacij, temveč tudi bistveno sredstvo komunikacije. Omogoča nam boljše življenje, izvajanje kakršnih koli transakcij in dejavnosti, medsebojno organizacijo v družbi in oblikovanje političnih mnenj. Programska oprema je ključna za delovanje gospodarskih, družbenih in političnih organizacij, ne glede na to, ali so javnega ali zasebnega značaja oziroma delujejo na mobilnih napravah ali v oblaku. Igra nepogrešljivo vlogo posrednika, ki omogoča dostop do vseh digitalnih informacij, poleg člankov in podatkov pa predstavlja tudi enega od stebrov sodobnega raziskovanja (Noorden et al., 2014).

Programska oprema je torej *pomemben del* naše znanstvene, tehnične in industrijske dediščine.

Ob pozornem opazovanju lahko ugotovimo, da se pravo znanje hranjeno v programski opremi ne skriva v izvršljivih programih, temveč v »izvorni kodi«, ki v skladu z definicijo v licenci GPL predstavlja »zaželeni način, s katerim razvijalec spremeni program.«[[1]](#footnote-1) Izvorna koda je posebna oblika znanja: ustvarjena je tako, da jo *razume človek*, torej razvijalec, in da se jo lahko strojno prevede v obliko, s pomočjo katere je *izvršljiva* neposredno na napravi. Zgovorno je že samo izrazoslovje, ki ga uporablja računalniška skupnost: »programski jeziki« se uporabljajo za »pisanje« programja. Harold Habelson je že leta 1985 zapisal, da je »programe treba najprej napisati, da jih lahko preberejo drugi ljudje«. (Abelson & Sussman, 1985)

Izvorna koda programske opreme je torej *človeška stvaritev, ki je ustvarjena na povsem enak način* kot drugi pisni dokumenti, zato si razvijalci programske opreme zaslužijo enako mero spoštovanja kot drugi ustvarjalci.

Izvorna koda programske opreme je dragocena dediščina, kar sta trdila že Len Shustek (v odličnem članku iz leta 2006 (Shustek, 2006)) in Donald Knuth (Knuth, 1984), zato je bistvenega pomena, da jo poskušamo ohraniti.

Poslanstvo pobude Software Heritage, ki je nastala leta 2015 s podporo inštituta Inria,[[2]](#footnote-2) je *zbirati, organizirati in ohranjati* vso javno dostopno *izvorno kodo* na planetu ter *jo narediti javno dostopno*, ne glede na to, kje in kako je bila razvita ali distribuirana.

* 1. *Večplastna naloga*

Arhiviranje celotne izvorne kode, ki je dostopna, je večplastna naloga. V članku (Abramatic et al., 2018) je podrobno opisano, da je treba uporabiti različne strategije, odvisno od tega, ali želite zbrati odprto ali lastniško izvorno kodo, poleg tega pa je treba izvorno kodo, ki je že dostopna prek spleta, drugače obravnavati kot izvorno kodo, ki je nameščena na starejših fizičnih medijih.

Za zbiranje odprte izvorne kode, ki je že dostopna na spletu, je najustrezneje razviti zbiralnik, ki samodejno zbira vsebino z najrazličnejših platform za sodelovanje, kot so GitHub, GitLab.com ali BitBucket, ali s platform za distribucijo paketov programske opreme, kot so Debian, NPM. CRAN ali Pypi.

Za zbiranje izvorne kode stare programske opreme je treba vzpostaviti pravi postopek računalniške arheologije. V sodelovanju z Univerzo v Pisi in UNESCOM smo že pričeli z delom in tako je nastal proces [SWHAP](https://www.softwareheritage.org/swhap). Uporabili smo ga za iskanje, dokumentiranje in arhiviranje programske opreme, ki je igrala pomembno vlogo v zgodovini računalništva v Italiji. Nedavno smo ga razširili v sklopu projekta [Software Stories](https://stories.softwareheritage.org), ki poskuša izpostaviti vse zgodovinske elemente v zvezi s programsko opremo, katere izvorna koda je bila najdena.

* 1. *Svetovno poslanstvo*

Temeljna načela Software Heritage so (Abramatic et al., 2018; Di Cosmo in Zacchiroli, 2017): sistematična uporaba odprtokodne programske opreme za razvijanje infrastrukture za Software Heritage, zato da je njeno delovanje mogoče razumeti in ga po potrebi ponoviti; izgradnja globalne mreže neodvisnih zrcalnih kopij arhiva, saj je veliko število kopij najboljša zaščita pred izgubo in napadi; izbira neprofitne in mednarodne strukture z več deležniki, s čimer se zmanjša tveganje za obstoj posameznih mest okvar ter zagotovi, da bo pobuda Software Heritage resnično služila vsem.

Za izpolnitev takega poslanstva sta potrebni institucionalna legitimnost kot tudi resnična sposobnost širokega konsenza. Okvirni sporazum, ki sta ga Inria in UNESCO podpisala 3. aprila 2017 in obnovila novembra 2021, je ključnega pomena.

* 1. Preteklost, sedanjost in prihodnost: veliko več kot arhiv!

Software Heritage poseduje infrastrukturo, ki raste iz dneva v dan. Arhiv večinoma sestavljajo vsebine, ki so bile pridobljene s postopkom samodejnega zbiranja, pričenjamo pa dodajati tudi bisere, pridobljene s potrpežljivim obnavljanjem starejše programske opreme s postopkom zajemanja, ki je bil razvit v sodelovanju Univerze v Pisi in UNESCA.[[3]](#footnote-3)

*Slika 1: Število projektov, izvornih datotek in različic, ki so bili od junija 2022 arhivirani v* [*Software Heritage*](https://www.softwareheritate.org/archive)*.*

Čeprav izčrpnost še zdaleč ni dosežena, arhiv že sedaj vsebuje največji korpus izvorne kode na svetu z več kot 180 milijoni arhiviranih izvorov za več kot 12 milijard edinstvenih datotek z izvorno kodo, vsaka od njih pa je opremljena z intrinzičnim identifikatorjem, ki temelji na kriptografskih zgoščenih vrednostih (Di Cosmo et al., 2018).

Ta edinstvena infrastruktura ima več poslanstev. Seveda je namen ohraniti izvorne kode iz *preteklosti*, ki so zapisale zgodovino računalništva in informacijske družbe, za prihodnje generacije. Vseeno pa predvsem poskušamo zgraditi *velikanski teleskop*, ki nam bo pomagal pri raziskovanju *sedanjega* razvoja *galaksije programske opreme*, da bi jo bolje razumeli in izboljšali ter tako ustvarili boljšo tehnološko *prihodnost.*

* 1. *Strateško vprašanje, ki ga moramo poznati*

Arhiv Software Heritage je že sedaj najpomembnejša zbirka izvorne kode na svetu, a pred nami je še dolga pot: združiti je treba še številne deležnike, od kulturne dediščine do industrije in od raziskovalnih ustanov do javne uprave. Računamo, da bomo to dosegli s pomočjo rastoče mreže [ambasadorjev](https://softwareheritage.org/ambassadors/) vključno z Računalniškim muzejem v Ljubljani.

Jasno je, da je programska oprema postala bistven sestavni del vseh človekovih dejavnosti, zato neomejen dostop do javno dostopnih izvornih kod programske opreme postaja vprašanje digitalne suverenosti vseh narodov.

Edinstvena infrastruktura, ki jo z univerzalnim pristopom gradi Software Heritage, je ključnega pomena za doseganje zahtevnega cilja digitalne suverenosti ob ohranjanju dimenzije arhiva za skupno dobro.

Zato je izjemno pomembno, da se odločevalci na ravni institucij, panoge, znanosti in civilne družbe pričnejo zavedati, kako pomembna so ta vprašanja. Poleg tega bi morala Evropska unija hitro zavzeti svoja stališče in pričeti zagotavljati potrebna sredstva, zaradi katerih bo lahko pobuda Software Heritage rasla in trajala. S tem bi se postavili ob bok drugim mednarodnim akterjem, ki so se k pobudi že zavezali, in podprli ustanovitev mednarodne neprofitne ustanove, ki bo dolgoročno izvajala to poslanstvo.

1. 2. Razvoj in zastaranje programov – izziv in nočna mora programerjev

Saša Divjak

Sam sodim v starejšo generacijo, saj sem svoje prve programerske izkušnje pridobil leta 1967 na legendarnem računalniku Zuse Z32. Začetek 70. let prejšnjega stoletja je bilo obdobje luknjanih kartic in perforiranega traku. Takrat je programiranje prvih mikro-, včasih pa tudi miniračunalnikov potekalo na zanimiv način. Kot periferno enoto računalnika smo uporabljali »Teletype«, ki je poleg tipkanja in tiskanja združeval še čitalnik in luknjač papirnega traku. Priprava in izvedba programa je potekala tako:

Najprej si moral v računalnik vnesti urejevalnik teksta, ki je bil pomnjen v binarni kodi na traku. Zatem si napisal svoj program in to izvorno kodo (pogosto v zbirnem jeziku) naluknjal na nov trak. Sledilo je branje zbirnika (angl. *assembler*), seveda spet binarno zakodiranega na drugem traku. In nato branje »svojega« programa v izvorni kodi. Kolikor se spomnim, je tudi to branje potekalo v dveh korakih, saj zbirnik za svoje delo potrebuje vsaj dve fazi. In končno, če je bilo (skoraj) vse v redu, si lahko naluknjal nov trak, tokrat s svojim programom v binarni kodi. Sledila sta branje binarno kodiranega novega programčka in njegova izvedba. Za to, kar danes naredimo v drobcu sekunde, si potreboval kar nekaj minut in upal, da se nisi kaj zmotil. Sicer si moral celoten postopek praktično ponoviti.

Pri programiranju mikroračunalnikov si svoj strojno zakodirani program običajno še »zapekel« v EPROM (integrirano vezje, ki je predstavljalo del pomnilnika mikroračunalnika). Tako smo razvijali različne mikroračunalniško podprte avtomatizacije. Programerji pa se radi motimo. Ker je celoten ciklus potreboval kar nekaj časa, smo pogosto (če se je dalo) popravke vnesli kar v strojno kodo in preskočili zamudno luknjanje traku. Posledično smo imeli na koncu pravilno delujoč program v EPROMU, izvorna koda pa temu ni več povsem ustrezala. Seveda je to zelo narobe.

Miselno prehajanje med programi v zbirnem jeziku in njihovo strojno kodo je bilo nekaj vsakdanjega. Nenazadnje smo v tistih časih pogosto vpisovali zagonski nalagalnik (t. i. *bootstrap loader)* kar prek stikal na konzoli računalnika. In, ker smo to pogosto počeli, ni bilo nič nenavadnega, da smo si na pamet zapomnili tudi zaporedja več 10 ukazov na strojnem nivoju. Take so bile na primer moje izkušnje s prvimi računalniki iz družine Digital PDP 11 v 70. letih prejšnjega stoletja.

Pred nadaljevanjem naj povem, da sem na Fakulteti za računalništvo in informatiko predaval predmete s področja programiranja, sistemske programske opreme in operacijskih sistemov. In, kar me je še posebej veselilo, računalniške grafike. To se odraža tudi v nekaterih spominih, ki jih bom navedel v nadaljevanju.

Danes je na svetu registriranih več kot 9.000 različnih programskih jezikov in tudi načini programiranja so se zelo spremenili. Na takratni Fakulteti za elektrotehniko in računalništvo na študijskem programu računalništva in informatike smo v 80. letih pri programiranju uvedli programski jezik C, ki je še danes »latinščina« in dobra osnova za marsikateri drugi programski jezik. V letu 1997 pa smo uvedli še programiranje v programskem jeziku Java. Kot nosilca predmeta me je vedno skrbel stalen razvoj tega, takrat še novega programskega jezika, ki je v naslednjih letih doživel kar nekaj bistvenih sprememb, saj smo aktualne jezike uporabljali tudi v okviru različnih projektov.

Po drugi strani smo pri razvoju spletnih aplikacij že v poznih 90. letih uporabljali še danes popularni JavaScript. Objektno usmerjenemu programiranju so se pozneje pridružili drugi pristopi. Med njimi je pomembno komponentno programiranje, ki pomeni uporabo različnih problemsko usmerjenih knjižnic in tako sestavljanje novih aplikacij s kombiniranjem lastne razvojne kode in funkcionalnih modulov, ki jih take knjižnice nudijo. Zakaj bi izumljali toplo vodo, če pa nekatere rešitve že obstajajo, na primer za podporo 2D in 3D računalniške grafike, pa za izvajanje različnih bolj ali manj kompleksnih, že preskušenih in računsko učinkovitih rutin. Tak pristop po eni strani predvideva poznavanje tako imenovanih API-jev (Application Programming Interfaces), hkrati pa zelo pospešuje razvojno delo. Vendar hkrati skriva tudi pasti, ki jih predstavlja razvoj novih in novih verzij, kar sčasoma nujno pripelje do neskladja posameznih komponent naših aplikacij in posledično zastarelosti našega programja. V tem smislu je posebej nevarno »mešanje« tehnologij različnih proizvajalcev, saj vsak sledi svojim standardom in usmeritvam. Kot primer naj navedem danes že praktično pozabljeni VRML (Virtual Reality Markup Language), ki se je pojavil sočasno s programskim jezikom JavaScript in je omogočal za tiste čase kar spodobno 3D-vizualizacijo in animacijo 3D-scen. Kombiniranje jezikov VRML in JavaScript je omogočalo izdelavo zelo atraktivnih 3D-vizualizacij in interaktivnih simulacij naravoslovnih pojavov. Danes so ti primeri že povsem zastareli in jih ne moremo več kazati (res pa je, da je današnja 3D-grafika vse kaj drugega). Spomnimo se še javanskih apletov, ki so omogočali raznovrstne aplikacije (in tudi 3D-vizualizacije) v naših brkljalnikih. Pa so ugotovili, da ima taka tehnologija preveč varnostnih lukenj, saj je omogočala tudi delovanje izven oglaševanega varnega »peskovnika«. Ponudniki brkljalnikov so to začeli drug za drugim onemogočati v novejših verzijah. Danes si lahko to ogledamo le na računalnikih, ki imajo v ta namen nameščene za današnje razmere zastarele operacijske sisteme in brkljalnike. Pri tem si pogosto pomagamo z »virtualkami« (navideznimi napravami) na našem računalniku. Razvijalci smo nujno iskali rešitve v sorodnih tehnologijah v upanju, da bomo sedaj pa imeli mir. Tako je marsikdo »preklopil« na nekdaj popularni Flash in na njegov programski jezik ActionScript. V nekaterih primerih je lahko tako ohranil tudi do 80 % svoje programske kode. Kasneje smo lahko ugotovili, da smo se »ušteli«. Danes je Flash (in s tem ActionScript) že izumrl. Prava usmeritev je bila v JavaScript oziroma v jQuery zaradi večje programerske učinkovitosti. Spet si običajno pomagamo z različnimi knjižnicami za učinkovito in poenoteno pripravo grafičnih uporabniških vmesnikov ipd.

Dandanes se srečujemo s številnimi mobilnimi napravami, od pametnih telefonov prek tablic do večjih ali manjših prenosnih in namiznih računalnikov. Zaradi raznolikosti zaslonov se je uveljavil »odzivni dizajn« (angl. *responsive design*), ki omogoča enako uporabniško izkušnjo na različnih napravah ne glede na velikost in resolucijo našega zaslona. Kako si lahko pomagamo pri razvoju takih aplikacij? Nekako logičen korak se zdi uporaba knjižnice jQuery Mobile, ki omogoča načrtovanje grafičnih uporabniških vmesnikov, ki so primerni za različne naprave. No, obstajajo tudi druge možnosti, na primer popularni Bootstrap. Vse lepo in prav, a le nekaj časa. jQuery je iz verzije 1 prešel v naslednje verzije (v času pisanja tega prispevka obstaja verzija 3). In razvojna orodja kar naenkrat razvijalcu pošiljajo opozorila (angl. *warning*), da so delci naše kode zastareli (angl. *obsolete, deprecated*). Če drugega ne, je to grdo in nas mora skrbeti, saj postaja naša koda stara in bo verjetno sčasoma neuporabna. Dobro, pa bomo izvedli migracijo (spremembo) naše kode, da bo ta skladna z novimi napotki. Posebno pri daljših programih se nam to zamudno delo upira, saj terja sistematičnost, a je nujno, če želimo slediti razvoju. A tu se skriva še kakšna dodatna past. Tako se je na primer razvoj pri jQuery Mobile ustavil in novim različicam jQuery ni več sledil. Preostane nam še dodaten prehod oziroma opustitev kode, ki jo predstavlja taka zastarana knjižnica.

Danes govorimo o izumrlih jezikih in delo programerja se ne zaustavi pri razvoju morda atraktivne aplikacije. Njeno vzdrževanje z leti terja dodaten napor.

Eno od vprašanj, ki se programerju poraja, je, komu sploh zaupati in slediti, da ne bo njegovo vloženo delo prehitro zastaralo. Gotovo so to »veliki svetovni igralci«, pa še tu se včasih zatakne. Če se povrnem na 3D-grafiko, se spomnim Microsoftove tehnologije Silverlight, ki je nudila možnost zelo lepih vizualizacij 3D-svetov in njihovih animacij. Past, ki se je pri tem skrivala, je bila ta, da je bil to pač Microsoftov izdelek. Ali mu bodo sledili tudi drugi? Poleg MS Windows imamo tudi druge operacijske sisteme. In res, Silverlight se ni prijel in je danes opuščen.

In tudi sedaj se razvijalci srečujemo s podobnimi dilemami. Nedvomno je zaradi popularnosti mobilnih naprav zanimiv razvoj aplikacij za operacijske sisteme Android in iOS. Kar nekaj razvijalskih platform najdemo na Internetu. Kateri naj zaupamo, katera se bo obdržala vsaj malce dlje? Katera pa bo hitro izumrla? Je to React? Ali pa morda Flutter, ki spet temelji na novem jeziku (Dart)? Le zakaj je to potrebno? Beremo forume, ustvarjamo svoje vtise in upamo, da gremo v pravo smer.

Če nek projekt začenjamo na novo, si moramo prej dobro ogledati, kateri trendi so danes uveljavljeni. Prevladujejo spletne aplikacije in čedalje bolj računalništvo v oblaku. Že kar nekaj časa je uveljavljena uporaba jezikov HTML5, pa CSS in AJAX/JSON. Jezik JavaScript (ali bolje JQuery) je izpodrinil Flash. Apleti so že davno izumrli.

Tudi načini razvoja programov so se spremenili. Uveljavljajo se metode hitrega inkrementalnega razvoja, ki zamenjujejo sekvenčni pristop. V sklopu istega projekta včasih uporabljamo po več jezikov in poznati moramo različne aplikacijske programske vmesnike (API). Kompleksnost programov se povečuje.

Zaznamo lahko polarizacijo programiranja: po eni strani uporabljamo visokonivojske programske jezike, ki omogočajo večjo produktivnost programerjev, pa tudi paralelizacijo in delo v oblaku. Po drugi strani so včasih pomembni učinkovitost kode in večanje hitrosti izvajanja ter uporaba asimetričnega računanja (tudi zaradi večjedrnih sistemov). Hkrati zasledimo »demokratično« računalništvo, ki pomeni, da si lahko kaj malega sprogramira ali vsaj priredi tudi vsak manj poučen (a motiviran) uporabnik. In omenimo še »nevarno računalništvo«. Če nekaj postane prekompleksno, si pač omislimo novo ogrodje (angl. *framework*), ki morda nadgrajuje nekaj prejšnjega. Stvar se kopiči podobno kot sklad umazane posode ali krožnikov in v sebi skriva neučinkovitost kode, pa tudi kakšne varnostne luknje.

Sprašujemo se lahko, kaj je še pred nami. Še vedno se moramo spomniti na znani Moorov zakon, ki govori o tem, da se število tranzistorjev (torej gostota integriranih vezij) podvaja vsaki dve leti. A kaj, ko obstajajo tako imenovani štirje Nathanovi zakoni, ki govorijo o tem, kaj se dogaja s programi. Že davnega leta 1997 jih je postavil Nathan Myhrvold, nekdanji glavni tehnološki direktor pri Microsoftu. Pri teh zakonih dobimo zanimivo asociacijo na Newtonove zakone, a le poimensko. Oglejmo si jih:

1. Nathanov zakon: Programje je kot plin.

Programska oprema se vedno razširi tako, da na koncu zasede katerikoli vsebnik (torej kapaciteto računalnika), v katerem je shranjena. Primere take rasti zasledimo pri novih in novih verzijah operacijskih sistemov, kot sta Windows in Linux, pa tudi pri razraščanju kode brkljalnikov.

2. Nathanov zakon: Programska oprema raste, dokler je ne omeji Moorov zakon.

Sprva se širi hitro, podobno kot se širi plin. A rast neizogibno upočasni zmožnost aparaturne opreme. Zato sleherni procesor prej ali slej poklekne. To se običajno zgodi malo pred pojavom novih modelov.

3. Nathanov zakon: Rast programov omogoča Moorov zakon.

Zato neprestano kupujemo nove računalnike. Integrirana vezja v njih so čedalje hitrejša, a cena računalnikov se bistveno ne spreminja. Več dobimo za isti denar. Ta fenomen se ponavlja, ker se pojavljajo novi in novi programi.

4. Nathanov zakon: Programe omejujejo le človeške ambicije oziroma pričakovanja.

Nikdar nam ni dovolj. Srečujemo se z novimi aplikacijami. In novimi pogledi, kaj je moderno.

Programi in tudi samo programiranje je tako venomer v krizi. Kar koli dosežemo, običajno ne dosega pričakovanj uporabnikov. Lestvica pričakovanj se neprestano dviguje.

Programiranje je umsko zahtevno tudi za izkušene programerje. Stalno preslikavamo med različnimi miselnimi modeli v naših glavah in pretvarjamo različne predstavitve v kodo in nazaj. Programi so pravzaprav abstrakcije in pri njihovem razumevanju si pogosto pomagamo s konkretnimi primeri.

Pomagamo si s pristopi, kot je objektno usmerjeno programiranje, in z uporabo višjih programskih jezikov. Pogosta praksa je tudi pristop kopiraj/prilepi, ki temelji na uporabi »izrezkov kode« (angl. *code snippets*).

Vemo, da še otroci bolje razumejo nekaj, kar vidijo, v primerjavi s tem, kar le slišijo. Kar čutimo, takoj razumemo, besede pa moramo najprej analizirati, če želimo vedeti njihov pomen. To ni problem pri kratkih stavkih. Dolga besedila pa terjajo več časa in napora. Tako je tudi s programsko kodo. Nekateri izvorni programi so težko razumljivi in včasih je enostavnejše, da jih razvijemo kar znova.

Bo kar držalo, da so programski jeziki namenjeni ljudem in ne računalnikom in da smo še v zgodnjem obdobju zgodovine programiranja. Velik premik je predstavljal prehod od luknjanih kartic in trakov na interaktivno delo za računalniškimi zasloni. Naslednji premik pa bo moral upoštevati, da v nekaj desetletjih pričakujemo razvoj tako sposobnih računalnikov, ki bodo dosegli inteligenčno raven ljudi. Le kako bomo take računalnike programirali? Ali se bodo takih veščin učili sami? Si bodo sami izmišljali nove standarde? Bodočnost bo vznemirljiva. Morda pa nas mora skrbeti.

1. 3. Od BESS do EVA, arhivski izziv

Primož Jakopin

* 1. 1. Uvod

1. Slovenski Računalniški muzej je ob zelo primernem času podal pobudo za organizacijo okrogle mize o dediščini informacijskih orodij (izključno naprednih), katere najpomembnejši del je programje. Leta 1960 je izšel prvi glasnik računalniške dobe v Sloveniji, in sicer knjiga profesorja Franceta Križaniča z naslovom *Elektronski aritmetični računalniki*, ki je temeljila na njegovih izkušnjah pri delu z računalnikom [Ural-1](https://en.wikipedia.org/wiki/Ural_(computer)) na Državni univerzi v Moskvi (MGU). Leta 1962 je v Sloveniji začel delovati prvi računalnik [Zuse Z23](https://en.wikipedia.org/wiki/Z23_(computer)) in na srečo je z nami še kar nekaj njegovih uporabnikov, ki lahko prispevajo svoje izkušnje iz prve roke.

V članku so prikazane avtorjeve izkušnje na njegovi poklicni poti po prehodu z velikih računalnikov na osebne mikroračunalnike, pa tudi področje urejevalnikov besedil z orodji za jezikovne tehnologije in korpusno jezikoslovje, vključno z leksikalnim spletnim iskalnikom. Predstavitvi avtorjevega dela je na koncu dodan tudi avtorjev predlog za razvoj spletnega muzejskega eksponata v obliki rešitve, ki bila zasnovana na podoben način kot vikipedija.

* 1. 2. STRUCTRAN

1. Ob koncu šestdesetih let in zlasti v zgodnjih sedemdesetih letih 20. stoletja je postalo jasno, da so programski jeziki tistega časa, kot sta na področju znanosti in tehnologije predvsem [FORTRAN](https://en.wikipedia.org/wiki/Fortran), v poslovnem okolju pa [COBOL](https://en.wikipedia.org/wiki/COBOL), manj primerni za vse kompleksnejše projekte. Dolgi in zapleteni algoritmi so imeli za posledico orjaške količine izvorne kode, ki jo je bilo vedno težje posodabljati in vzdrževati. Glavni razlog težav je bilo to, da so bili tedanji programski jeziki bližje strojni kodi in niso bili dovolj abstraktni oz. strukturirani. Profesor Edsger Dijkstra je leta 1968 objavil pismo z naslovom [*Stavek Go To je škodljiv (angl. Go To Statement Considered Harmful)*](https://en.wikipedia.org/wiki/Considered_harmful). Zaradi preskakovanja gor in dol po kodi se hitro izgubi nit, kaj je hotel programer, tudi njemu samemu, še posebej nekaj tednov ali mesecev pozneje. Ta zamisel je vodila do razvoja strukturiranega programiranja, torej kodiranja brez stavkov Go To, ki so jih nadomestili s krmilnimi strukturami z začetkom in koncem, kot je IF ELSE ENDIF, ter z zankami, kot so FOR ENDFOR, REPEAT ENDREPEAT, WHILE ENDWHILE.

Ker je razvijanje prevajalnika zelo dolgotrajno in zamudno, je Vladimir Batagelj z Oddelka za matematiko na Fakulteti za naravoslovje in tehnologijo [Univerze v Ljubljani](https://en.wikipedia.org/wiki/University_of_Ljubljana) predlagal razvoj predprocesorja STRUCTRAN, ki bi strukturirano kodo prevedel v standardni FORTRAN, njegov prevajalnik pa v izvršljivo datoteko po običajnem postopku. Vladimir Batagelj in Egon Zakrajšek sta zamisel leta 1975 opisala v članku z naslovom [STRUCTRAN](http://vladowiki.fmf.uni-lj.si/lib/exe/fetch.php?media=vlado:pub:conf:structran.pdf). Projekt naj bi izvedli študentje pri predmetu programiranje v šolskem letu 1975/76. Ker rezultati niso bili uporabni, je Egon Zakrajšek v 14 dneh pred koncem šolskega leta razvil končno različico kode, in sicer na računalniku Control Data Cyber 72, takrat največjem in osrednjem računalniku v Sloveniji, med njegovimi uporabniki je bila tudi Univerza v Ljubljani.

V spodnjem primeru je desni stolpec izvzet iz prispevka Batagelja in Zakrajška – opombe so izpuščene, da je besedilo čim krajše in jasnejše. Gre za [najenostavnejši](https://en.wikipedia.org/wiki/Bubble_sort) podprogram za razvrščanje seznama celih števil, ki so vsebovana v nizu tabeli TAB dolžine LENGTH. Kot pri programskem jeziku FORTRAN so vse spremenljivke z imeni, ki se začnejo s črkami I, J, K, L, M in N, cela števila, razen če ni določeno drugače – vnaprej so deklarirane le spremenljivke BOUND, CHANGE in TAB. Različica v programskem jeziku FORTRAN (levi stolpec) ne vsebuje stavkov Go To, preskoki so izvedeni s stavki IF. Izrazita premoč strukturiranega programiranja bi morala biti očitna tudi osebam, ki s programiranjem nimajo izkušenj.

Tabela 1: Podprogram za mehurčno razvrščanjev programskih jezikih FORTRAN in STRUCTURAN

STRUCTRAN je bilo prvo splošno uporabljeno programje, ki je bilo razvito v Sloveniji. Mnogim programerjem je precej olajšalo poklicno življenje, med drugim tudi avtorju tega prispevka.

* 1. 3. Od BESS do EVE EVA

[Mikroračunalniška revolucija](https://en.wikipedia.org/wiki/History_of_personal_computers) se je v 70. letih začela na drugi strani Atlantika in večino starega sveta je v vsej svoji razsežnosti dosegla v zgodnjih 90. letih. Čeprav so prvi osebni računalniki prišli do, oprostite izrazu, širokih ljudskih množic, predvsem kot stroji za igrice, so računalniški strokovnjaki v njih videli povsem nekaj drugega, možnost, da bi računalnik postavili na svojo mizo, torej ne v obliki [terminala](https://en.wikipedia.org/wiki/Computer_terminal), prek katerega bi lahko dostopali do osrednjega računalnika, ki bi si ga delili z množico drugih uporabnikov. Leta 1981 oz. na začetku leta 1982 je Saša Albert, kolega iz [Narodne in univerzitetne knjižnice](https://sl.wikipedia.org/wiki/Narodna_in_univerzitetna_knji%C5%BEnica) in vsestranski tehnični mojster, prišel do mikroračunalnika [Sinclair ZX81](https://en.wikipedia.org/wiki/ZX81) z 1 kB spomina, snemalnikom [kaset](https://en.wikipedia.org/wiki/Cassette_tape) kot zunanjim pomnilnikom in [interpreterjem (sprotnim prevajalnikom) za BASIC](https://en.wikipedia.org/wiki/BASIC_interpreter). Avtorju prispevka, ki je bil v tistem času znan programer, ga je za nekaj dni posodil, da bi slednji preveril, ali bi se dalo s tem strojčkom narediti kaj uporabnega. Kot monitor je uporabil svoj manjši TV-sprejemnik. Koda v BASICu za izračun koordinat x, y in z jamskega poligona (jamarji merijo jame tako, da po tleh postavijo točke in med njimi merijo razdalje, azimute in naklone)CAVE SURVEY POINTS, ki je bila hitro zapisana na papir, je vsega skupaj obsegala en list A4 in še približno deset vrstic na drugem list. Prvo stran je bilo še mogoče shraniti v pomnilnik, naslednjih desetih vrstic pa ne. Saši je avtor napravo vrnil z oceno: z računalnikom bi se dalo narediti kaj praktično uporabnega, vendar bi potrebovali vsaj 2 kB spomina.

Naslednji računalnik je bil [Sinclair ZX Spectrum](https://en.wikipedia.org/wiki/ZX_Spectrum) z 48 kB spomina. Franci Ambrožič, kolega s Fakultete za telesno kulturo, je imel v Veliki Britaniji znance, ki so mu julija leta 1982 v njegovem imenu naročili računalnik. Povpraševanje je bilo tako veliko, da je izvozna cena, ki jo je Ambrožič moral plačati, znašala 228 £ v primerjavi z britansko domačo ceno 175 £, poleg tega pa je za tuje kupce veljal skoraj šestmesečni dobavni rok. Decembra 1982 so računalnik pretihotapili iz Anglije v Jugoslavijo, kjer ga je Franci Ambrožič z veseljem uporabljal nekaj časa, nato pa ga je med novoletnimi prazniki tako kot Saša Albert posodil avtorju prispevka, da bi presodil, ali bi ga lahko uporabili za kaj konkretnega. Tokrat je bil odgovor pritrdilen. Ker naprava ni imela urejevalnika besedil, ga je razvil kar avtor sam. V približno enem tednu je spravil skupaj urejevalnik besedil BESS (angl. Basic Editor for the Sinclair Spectrum). V celoti je bil napisan v programskem jeziku BASIC, na zaslonu pa je bilo prikazanih 24 vrstic po 32 znakov. Interpreterska narava programa BASIC je bila najbolj opazna med iskanjem po besedilu, saj je trajalo približno eno minuto na stran (2 kB). Ker zbirni jezik še ni bil na voljo, je avtor napisal podprogram za iskanje po besedilu kar neposredno v strojni kodi, dolg je bil 50 bajtov, v 0,05 sekunde pa je preiskal 20 kB besedila. V letih 1983 in 1984 je zasnoval urejevalnik besedil TESS (urejevalnik besedil za računalnik Sinclair Spectrum, angl. *Text Editor for the Sinclair Spectrum*), 80 % ga je bilo že v zbirnem jeziku mikroprocesorja Zilog Z80. Leta 1985 pa je nastal urejevalnik INES (angl. INformation Editing System), ki je bil na voljo za široko uporabo predvsem v Sloveniji, pa tudi v drugih delih Jugoslavije. Skoraj v celoti je bil napisan v zbirnem jeziku in podpiral je datoteke do dolžine 21 kB. V urejevalniku je bilo 24 vrstic po 64 znakov (vsak v matrici 5 x 8 pikslov), imel je nastavitev za bele znake na črnem ozadju (ali obratno), tako da je le 8 % TV-zaslona oddajalo škodljive žarke, in kurzor, ki ni utripal. Poleg tega na zaslonu niso bile prikazane stalne moteče informacije, imel je gladko drsenje vrstic po zaslonu in nova stran besedila je prepisala prejšnjo (ni bila prikazana na novem praznem zaslonu). Program INES je lahko vrstice z besedilom obravnaval kot zapise podatkov, pri čemer je omogočal razvrščanje, iskanje, sestavljanje dopisnih seznamov (angl. mailing list) in še nekaj drugih možnosti obdelave podatkov. Črno-bele slike so bile v besedilo vključene kot ubežne sekvence. Na voljo je bil slovenski uporabniški priročnik na 68 straneh, za urejevalnik pa so bili na voljo še dodatne programi, npr. za vnos podatkov in za razvrščanje večjih datotek do velikosti 37 kB. Program in priročnik sta bila prevedena v angleški in nemški jezik ter poslana računalniškim revijam v Nemčiji in Veliki Britaniji. V začetku leta 1986 je programu INES sledil program EVE (Event Editor), ki je omogočal še več funkcionalnosti, vendar je bil zanj narejen le kratek seznam ukazov, na nekaj straneh. Nadaljnji razvoj je bil ustavljen, ker se je medtem pojavil [računalnik ATARI ST](https://en.wikipedia.org/wiki/Atari_ST) s čudovitim, neprierno močnejšim procesorjem [Motorola 68000](https://en.wikipedia.org/wiki/Motorola_68000), ki je imel šestnajst 32-bitnih registri.

Program EVE je nasledil STEVE (ST EVent Editor, za Atari ST). Njegovo ime je bilo kot pri predhodnih programih tudi v obliki kratice, vendar tokrat drugega spola. Tako kot pri [imenih ladij](https://catamaranguru.com/why-are-boats-named-after-women/) je tudi tukaj šlo za izjemo od pravila, BESS, TESS, INES, EVA in NEVA - EVA je bila EVE za PC-kompatibilne računalnike. Pri programu INES se je avtor spomnil še ene znane izjave profesorja Djikstra, in sicer: [»Mikroračunalniki niso dobri (angl. *Microcomputers are not great*).«](https://www.cs.utexas.edu/users/EWD/transcriptions/EWD06xx/EWD634.html) (IFIP 1977) Da bi kar najbolje izkoristili naprave, ki jih poganjajo mikroprocesorji (da bi hitrost programja povečali vsaj za trikrat), so morali programerji tu programirati v strojnem jeziku. Program STEVE je bil iz 75.000 vrstic takšne kode, ki je registre obravnavala neposredno, brez elegance in preprostosti strukturiranega programiranja.

Slovenska različica programa STEVE je bila skupaj s priročnikom na 248 straneh pripravljena leta 1986 (drugi priročnik na 290 straneh je leta 1989 napisal Hinko Muren). Nato sta leta 1987 sledila hrvaški in nemški prevod (nemškega je prevedel Günther Weber), drugi nemški priročnik sta leta 1988 pripravila Klaus Detlef Olof in Peter Wieser (na 356 straneh), dokončni priročnik [STEVE Reference Manual](https://jakopin.net/steve/STEve_Reference_Manual.php) na 608 straneh pa je v angleščini izšel leta 1989. Računalnik Atari ST je z več megabajti linearnega spomina, disketo in trdim diskom ter odličnim monitorjem odpiral nova obzorja in omogočal uresničevanje idej, ki so presegale zmogljivosti osrednjih računalnikov. Program STEVE je prinesel kar nekaj novosti. Med drugim so bili vsi njegovi viri, tj. od sporočil na zaslonu, postavitve tipkovnice, pisav na zaslonu in tiskalniku do okrajšav ukazov, shranjeni v ločeni datoteki virov z imenom STEVE.RSF, ki jo je bilo mogoče urejati z urejevalnikom samim, s programom STEVE. Če omenimo še nekaj drugih lastnosti: velikost datoteke je bila omejena le z razpoložljivim pomnilnikom, vrstice urejevalnika so lahko vsebovale besedilo ali črno-belo grafiko, na voljo sta bila dva grafična urejevalnika, zbirke podatkov je znal STEVE obdelovati tudi kot verige datotek, kjer je edino omejitev predstavljala velikost prostora na disku. Obstajali so tudi lastno namizno založništvo, računalniško podprte učne ure (angl. Computer Aided Instruction) in tudi lasten [grafični uporabniški vmesnik](https://en.wikipedia.org/wiki/Graphical_user_interface).

STEVE je bil na razpolago v Jugoslaviji, Nemčiji, Avstriji, Švici, državah Beneluksa in Skandinaviji.

V zgodnjih 90. letih je znamka ATARI propadla, zato je moral avtor zamenjati računalniško platformo in pričel je pisati za [osebni računalnik tipa IBM PC)](https://en.wikipedia.org/wiki/IBM_PC_compatible). V tistem času so bili osebni računalniki zasnovani na osnovi mikroprocesorja [Intel 80286](https://en.wikipedia.org/wiki/Intel_80286), ki je bil neprimerno slabši od Motorole 68000. Avtorju se je to zdelo zelo žalostno in izkazalo se je, da ga tudi ni bilo smiselno programirati v strojnem jeziku. Primeren operacijski sistem za uporabo na osebnih računalnikih z grafičnim uporabniškim vmesnikom za PC (Windows 95), primerljiv z računalniki Atari ST ali [računalniki Apple Macintosh](https://en.wikipedia.org/wiki/Macintosh), je bil na voljo šele leta [1995](https://en.wikipedia.org/wiki/Windows_95).

Čeprav bi prenos programa STEVE in njegove obsežne baze strank na osebni računalnik v povsem nespremenjeni obliki predstavljal takrat najboljšo poslovno idejo, pa ljudje, zlasti razvijalci, vedno poskušajo ustvariti nekaj novega, boljšega in bolj vznemirljivega. Zato se je avtor na podlagi svojih dotedanjih izkušenj odločil narediti nov in močno izboljšan urejevalnik, v katerem je obdržal vse uporabne vidike programa STEVE in dodal nova orodja, ki bi jih lahko dobro izkoristil v nadaljnjem poklicnem življenju, potem ko bi prenehal delati kot samostojni razvijalec. Novi program, ki ga je poimenoval EVA, tokrat ni bil v obliki kratice, EVA kar tako. Leta 1992 se je najprej imenoval EVA za DOS, za operacijski sistem Windows pa je bil prilagojen leta 1996. Napisan je bil v programskem jeziku C tako kot tudi primeri v knjigi avtorja [Charlesa Petzolda](https://en.wikipedia.org/wiki/Charles_Petzold) z naslovom *Programming Windows 95*. S programskim jezikom C++ se avtor ni mogel dovolj približati računalniku, da bi lahko izvedel vse, kar je želel, predvsem pri uporabniškem vmesniku. Program EVA ima preprostejši grafični urejevalnik, je brez računalniško podprtih učnih ur in ima precej omejeno namizno založništvo. Po drugi strani pa ima pravi 16-bitni nabor znakov z 2500 implementiranimi znaki, vrstice, ki lahko vsebujejo 8-bitno besedilo, 16-bitno besedilo, stisnjene črno-bele grafike in enovrstične bitne slike v obliki črno-belih grafik, razširjene možnosti obdelovanja podatkovnih zbirk, tudi za podatkovni format XML, optično prepoznavanje znakov (OCR) kot standard (pri programu STEVE je bilo to treba doplačati), oblikoslovno označevanje (angl. *POS tagging*) in mnoga druga jezikovnotehnološka orodja, kot na primer statistično modeliranje jezika. Ena od glavnih zanimivosti programa STEVE, ki je ohranjena v programu EVA, so velike enoširinske bele črke na temnem ozadju. Po njihovi zaslugi je avtor ohranil svoj vid, čeprav je računalnik uporabljal večino časa, med drugim tudi za izdelavo in posodabljanje spletnih strani. Mnogi njegovi kolegi take sreče niso imeli, če so njihovi poklici zahtevali celodnevno delo pred računalniškim zaslonom. Skoraj vsi zdaj nosijo očala in zdravniki so jim priporočili uporabo računalnika eno uro ali manj na dan.

V juniju 2022 je bila izvorna koda programa EVA dolga 158.211 vrstic, imela je 2.684 funkcij in podprogramov. Program STEVE ima 414 ukazov, program EVA pa 1.385.

* 1. 4. NEVA

Ko je avtor leta 1998 zaključeval svojo doktorsko disertacijo z naslovom [Zgornja meja entropije pri leposlovnih besedilih v slovenskem jeziku](https://www.jakopin.net/primoz/disertacija/izvlecek.php) na Fakulteti za elektrotehniko Univerze v Ljubljani, ga je njegov glavni mentor prof. Nikola Pavešić opozoril, da besedilo ne sme presegati 200 strani. Avtor je v disertaciji obravnaval besedilni korpus s tremi milijoni besed in v priloge je želel vključiti več zanimivih izvlečkov. To pa bi bistveno preseglo omejitev na 200 strani, zato so mu svetovali, naj priloge objavi na spletu. Predstavitev korpusa bi lepo popestrila možnost iskanja, zato se je avtor naučil uporabljati vmesnik [Common Gateway Interface](https://en.wikipedia.org/wiki/Common_Gateway_Interface) in njegov skriptni jezik. Napisal je iskalnik (angl. search engine) za strežniško platformo Windows, ki je prav tako napisan v programskem jeziku C. Gre za podmnožico programa EVA, ki vsebuje predvsem ustrezna podatkovna orodja z izboljšanim dopisnim seznamom za izdelavo spletnih strani z rezultati iskanja. Program je poimenoval NEVA (Networked EVA) in v juniju 2022 je imel 491 postopkov v 25.306 vrsticah kode.

Avtor se je po zaključenem doktoratu leta 1999 med zaposlitvijo na [Filozofski fakulteti](https://www.ff.uni-lj.si/) [Univerze v Ljubljani](https://sl.wikipedia.org/wiki/Univerza_v_Ljubljani) odločil, da bo osnovni korpus s tremi milijoni leposlovnih besedil v slovenščini, ki je bil nameščen na strežniku fakultete, povečal in vanj vključil dodatna besedila, ki so bila večinoma iz časnika [DELO](https://en.wikipedia.org/wiki/Delo_(newspaper)). Pri DELU so razvili e-poštno storitev za slabovidne (DELO za slepe), ki so vsako jutro prejeli na svoj računalnik izvod v obliki besedila. Z dovoljenjem za uporabo v raziskovalne in izobraževalne namene mu je uspelo velikost korpusa, ki se je zdaj imenoval CORTES (CORpus of TExts in Slovenian, leta 1999), v začetku leta 2000 povečati na 28 milijonov besed. Korpus je pritegnil pozornost [Inštituta za slovenski jezik Frana Ramovša ZRC SAZU](https://isjfr.zrc-sazu.si/sl/predstavitev) (ISJ), kjer je bil avtor zaposlen s polovičnim delovnim časom. Ker Filozofska fakulteta ni mogla zagotoviti institucionalnega okvira za korpus CORTES in ker sta obe ustanovi tudi sicer zelo povezani, avtorja ni bilo težko prepričati, da korpus prenese na strežnik [Raziskovalnega centra Slovenske akademije znanosti in umetnosti](https://zrc-sazu.si/sl), katerega član je tudi ISJ. Maja leta 2020 je bil korpus preimenovan v Nova beseda, celotno spletno mesto pa v [BOS – Bank of Slovenian (Vsenarodni korpus besedil v slovenskem jeziku)](http://bos.zrc-sazu.si/index.html).

Avtor se je preselil na Inštitut, kjer je vodil novoustanovljeni Laboratorij za korpus slovenskega jezika, na Filozofski fakulteti pa je še naprej, s polovico delovnega časa poučeval jezikovne tehnologije.

*Slika 1: Spletni portal* [*http://bos.zrc-sazu.si*](http://bos.zrc-sazu.si)

Sčasoma so mnogi leksikalni viri Inštituta postali dostopni prek spletnega mesta BOS, nekateri izmed njih tudi v angleščini (leta 2015 je bilo vzpostavljeno novo spletno mesto [Fran](https://fran.si/) kot glavno slovarsko spletno mesto Inštituta v slovenščini). Najpomembnejši in najpogosteje uporabljeni so bili Slovar slovenskega knjižnega jezika (SSKJ, 1970–1991, 93.500 gesel), Besedišče slovenskega jezika (BSJ, 2006, 356.000 gesel) in besedilni korpus Nova beseda (2010, 318 milijonov besed). Medtem ko je SSKJ reprezentativni enojezični slovar slovenskega jezika, pa BSJ, ki ga sestavljajo SSKJ, po gradivu za slovar sodobnega knjižnega jezika zbrane besede, ki niso bile sprejete v Slovar slovenskega knjižnega jezika (178.457 enot), besedilni korpus Nova beseda in indeks slovenskega spletnega iskalnika [NAJDI.SI](http://www.najdi.si/), kaže živahnost in prilagodljivost slovenščine pri tvorjenju novih besed. Velikost slovarja BSJ je smiselna tudi v primerjavi s [številom besed v angleščini](https://www.merriam-webster.com/help/faq-how-many-english-words). Spodnja primera ponazarjata uporabo programa NEVA pri proučevanju dveh komponent imena (slovenskega) Računalniškega muzeja.

Tabela 2: Rezultat iskanja »muzea« ali »muzej« na seznamu slovenskih besed

Prikazanih je 32 besed, izmed katerih je 9 besed poudarjenih – te besede so navedene tudi v SSKJ. Izmed vseh teh besed je 20 samostalnikov, 10 pridevnikov in dva glagola. Zanimivo izstopata dve posebni vrsti muzejev, in sicer ekomuzej in fotomuzej.

Tabela 3: Rezultat iskanja »računal« na seznamu slovenskih besed

Prikazanih je 82 besed, izmed katerih je le 6 besed navedenih v SSKJ, saj je slovar nastajal do leta 1991, kar je bilo obdobje pred vzponom spleta. Nenavaden je le en vnos, in sicer obračunalec. Beseda bi lahko imela neračunalniški pomen, a se je po podrobnem pregledu izkazalo, da izhaja iz besednih iger.

* 1. 5. Arhivski izziv

Vsak muzejski eksponat, še posebej če so ga ustvarile človeške roke in srce, vzbudi številne občutke, vtise in misli, pa naj gre za [prvi kipec Venere](https://commons.wikimedia.org/wiki/File:Venus_of_Tan-Tan.jpg), [55.000 let staro piščal](https://en.wikipedia.org/wiki/Divje_Babe_flute) ali modernejši [uhan iz dragih kamnov](https://cdny.de/p/t/5/088/4469638.jpg). Kako ga je avtor izdelal? Kaj ga je pripeljalo do te odločitve? Kdo ga je uporabljal? Vsa ta vprašanja so še bolj izrazita pri računalniških eksponatih, vse od posameznih kosov strojne opreme in prvih znanilcev digitalne dobe do programja, katerega namen je čim boljše uporabiti starejše in tudi novejše kibernetske stroje.

Avtorjeve izkušnje z razvojem praktično istega izdelka bodo proti koncu tega leta štele 40 let. Počasi je ta urejevalnik besedil pod svoje okrilje vzel številna orodja, ki so avtorju pomagala rešiti nešteto nalog, s katerimi se je srečeval v poklicnem življenju in med obštudijskimi dejavnostmi. Prve različice, od BESS do INES in v manjši meri tudi STEVE, so kar najbolje izkoristile mikroprocesor računalnika. Pozneje je strojna oprema postala dovolj hitra, da je omogočala strukturirano programiranje v visokonivojskem jeziku, program pa je bilo lažje vzdrževati in dopolnjevati. Pri razvoju je bilo upoštevanih mnogo potreb in predlogov uporabnikov. Spremembe so se v prvem desetletju vrstile zelo hitro, drugo desetletje so zaznamovali vzponi in padci, v tretjem desetletju je bilo število sprememb zmernejše, v četrtem pa je do njih prihajalo le občasno. Koda zgodnjih različic programov BESS, TESS in INES je bolj ali manj izgubljena. Med pogostimi selitvami se je sled za številnimi starejšimi mediji izgubila, kasete Sinclair ZX Microdrive pa niso bile posebej zanesljive in bile so tudi zelo majhne. Programa STEVE in EVA za diskovni operacijski sistem DOS sta bila shranjena na disketah, ki so še vedno pri avtorju, vendar njihovo stanje ni znano. Format disket z računalnika ATARI ST, ni berljiv na osebnih računalnikih (PC), zato bi bila pomoč Računalniškega muzeja dobrodošla pri pripravi kataloga te zbirke. Izvorni referenčni priročnik STEVE v formatu STEVE DTP, ki ga je lahko bral tudi program EVA, se je na srečo ohranil, zato je bil med pripravami na okroglo mizo simpozija Računalniškega muzeja junija 2022 pretvorjen v obliko HTML in [objavljen na spletu](https://jakopin.net/steve/STEve_Reference_Manual.php). Izvorni kodi programov EVA in NEVA, ki sta bili razviti in vzdrževani v okoljih [Microsoft Visual C](https://visualstudio.microsoft.com/vs/community/) in [Bloodshed's Dev-C++](https://www.bloodshed.net/), še vedno hrani avtor.

Kako najbolje ohraniti to dediščino? Kako jo ustrezno predstaviti v Računalniškem muzeju? In kako eksponat prenesti v spletno okolje? Za začetek bi bila ena od možnosti vključitev v [spletni arhiv The Internet Archive Software Collection](https://archive.org/details/software), še bolje pa bi bilo, da bi ga ohranili v delujočem stanju. V nasprotju z večino naše digitalne zgodovine programa EVA in NEVA še vedno delujeta, sta v uporabi.

Vloga knjižničarja se je spremenila od osebe, ki je v preddigitalni dobi zainteresiranemu uporabniku povedala, kje in v kateri knjigi bo našel informacije, ki jih rabi, do osebe, ki uporabniku svetuje, kje na spletu bo našel tisto, kar potrebuje. Zato bi se morala spremeniti tudi vloga muzejskega kustosa in muzejskega vodiča. Z boljšim in poglobljenim znanjem o eksponatu programske opreme ga je obiskovalcem muzeja mogoče predstaviti na živahen in interaktiven način, ki je neprimerljivo boljši od opisa na spletu. Še toliko več, če programska oprema še vedno deluje, kot to na primer velja za programa EVA in NEVA. Takšen kustos bi bil tudi dragoceni vir informacij za osebe, ki morajo najti rešitev za problem, ki je bil verjetno že rešen, vendar rešitev ni splošno znana, je na spletu slabo dokumentirana ali pa sploh ne.

Takšno muzejsko predstavitev bi najlažje pripravila oseba, ki dobro pozna temo, najbolje kar avtor sam, in muzejski kustos, ki bi predstavitev izvedel. Ta postopek bi zagotovo zahteval kar nekaj časa, vendar bi bil ta čas dobro porabljen.

Predstavitev programske opreme v obliki spletne razstave je povezana tematika, a je vendarle drugačna. Čeprav je spletni arhiv [The Internet Archive Software Collection](https://archive.org/details/software) zelo pomemben, pa programske opreme ne predstavlja na enoten in muzejski način, tudi viri niso na voljo (odprtokodna platforma [SourgeForge](https://sourceforge.net/) ima dvakrat manjše število vnosov in ne gre za arhivsko zbirko). Da bi programa EVA in NEVA lahko predstavili v obliki spletnih muzejskih eksponatov, bi morali uporabiti medij, ki bi bil podoben spletnemu mestu [Wikisource](https://en.wikisource.org/wiki/Main_Page), vendar prilagojen predstavitvi programske opreme. V nasprotju s knjigami, kjer lahko bralec stran za stranjo uživa v umetniškem odsevu človeške domišljije, lahko delovanje programske opreme občudujemo ob obdelovanju niza podatkov, in sicer od majhnega števila, ki je seme za generator naključnih števil do urejevalnika besedil, ki lahko obdeluje [4 milijarde besed](https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia) v člankih na Wikipediji (toliko jih je bilo aprila 2022).

1. 4. Prehod na trajnostno uporabo programske opreme za dolgoročen dostop do digitalne dediščine

Nataša Milić-Frayling

V tem prispevku obravnavamo transformativni učinek uporabe digitalnih medijev na kodiranje in shranjevanje informacij ter pomen programske opreme za obdelavo digitalnih podatkov in prenos znanja. Zaradi hitrega razvoja inovacij na žalost hitro zastara tudi programska oprema, kar vpliva na našo zmožnost uporabe digitalnih vsebin. To je še posebej zahtevno pri zelo interaktivnih in dinamičnih digitalnih artefaktih, ki s programskim računanjem pridobivajo in posredujejo vpoglede na podlagi podatkov ter proučujejo pojave. Hitreje kot uvajamo inovacije, hitreje programsko opremo nadomeščamo z novimi izdelki. Programska oprema posledično ni več podprta in postane zastarela. Zato je težko ali nemogoče ponoviti pretekle analize podatkov, igrati stare igre in uporabljati interaktivne vsebine, kot je denimo digitalna umetnost. Na srečo nam napredek na področju računalništva prinaša tudi sredstva, s katerimi lahko preprečimo učinke zastaranja programske opreme. V družbi Intact Digital smo razvili platformo Software Library, ki po zaslugi virtualnih računalniških okolij omogoča stabilno in zaščiteno nameščanje podedovane programske opreme ter omogoča varno in enostavno uporabo digitalnih vsebin, ki so nastale pred desetletji. Ta pristop na učinkovit način omogoča digitalno kontinuiteto, ki je bistvenega pomena za prenos znanja na prihodnje generacije in nadgradnjo naše digitalne dediščine.

* 1. Digitalni mediji

Prihod digitalne revolucije v sredini 20. stoletja je prinesel najhitrejšo rast v nastajanju vsebin in informacij ter do tedaj najhitrejši prenos in izmenjavo podatkov in znanja prek računalniških tehnologij. Zdaj, 70 let pozneje, pa si sveta brez spletnih storitev in mobilnih naprav ne moremo več predstavljati, saj pomembno vplivajo na vse vidike našega vsakdanjega življenja.

Znanstvena in tehnična področja so prav tako doživela velike spremembe. Digitalne tehnologije, ki so vgrajene v instrumente in orodja, omogočajo neverjetno napredno analiziranje podatkov in odkrivanje znanja. S povečanjem zmogljivosti prenosnih naprav za shranjevanje podatkov in prehodom v računalništvo v oblaku in shranjevanje v oblaku, moramo danes pogosto skrbeti za terabajte osebnih vsebin ter petabajte znanstvenih in poslovnih podatkov. Kljub temu so digitalni mediji in digitalno računalništvo odvisni od zelo naprednih tehnologij, ki jih je treba nenehno posodabljati, da ostanejo funkcionalne in uporabne. Zato je izjemno pomembno, da upoštevamo tehnološke, ekonomske in izobraževalne dejavnike, ki vplivajo na kontinuiteto uporabe digitalnih medijev, in sprejmemo ukrepe, s pomočjo katerih bo naša digitalna dediščina živela še naprej in dosegla prihodnje generacije.

* 1. Digitalna kontinuiteta in pomen programske opreme

Da bi zagotovili dolgotrajno uporabo digitalnih vsebin, moramo razumeti glavne vidike digitalnih medijev ter načine njihovega staranja in propadanja.

Vsaka digitalna vsebina je ustvarjena, zbrana, shranjena in uporabljena s pomočjo združljive programske opreme. Digitalni mediji so v osnovi zato pravzaprav računalniške narave, saj različne funkcionalnosti programske opreme oblikujejo kodirane podatke in informacije. Da bi lahko ponovno uporabili plodove našega dela, digitalne dokumente, slike, videoposnetke in podatkovne zbirke shranjujemo kot datoteke. Vsako tako datoteko zapiše in prebere določena računalniški program ali vrsta združljivih programov, ki je zmožna obdelave takšnih datotek. Brez združljive programske opreme digitalno kodiranih vsebin ni mogoče interpretirati, predstaviti in doživljati. Pa vendarle tisti, ki se osredotočajo le na shranjevanje podatkov, pozabljajo na pomen programja.

*Digitalne datoteke brez programske opreme so kot partiture brez inštrumentov in glasbenikov. Shranjenih digitalnih vsebin ne bomo nikoli mogli predvajati in doživeti brez delujoče programske opreme in brez spretnosti, ki so potrebne za njeno uporabo.*

Čeprav so nekatere vrste programske opreme ustvarjene za uporabo z določeno strojno opremo, je večino programske opreme mogoče namestiti na različne vrste strojne opreme ali v navidezne naprave. Programsko opremo je mogoče namestit, če strojna oprema teče na kompatibilnemu operacijskemu sistemu. Čeprav je zastarelost strojne opreme prav tako problematična, se bomo za namen te razprave osredotočili na težave, ki izhajajo iz zastarelosti operacijskih sistemov in programske opreme.

Pri programski opremi gre lahko za aplikacijo, na primer urejevalnik besedil za ustvarjanje novih dokumentov, ali igro, ki jo zaradi skrbno zasnovane interaktivne izkušnje z veseljem igramo. Vsak program odvisen od številnih drugih tehničnih komponent, vse od programske opreme, ki omogoča uporabo miške, tipkovnice in zaslona, do varnostnih popravkov, ki zagotavljajo varnost operacijskega sistema in celotnega računalnika. Zato je treba neprestano spremljati in posodabljati vse programske aplikacije glede sprememb drugih podpornih komponent, da ostanejo funkcionalne in uporabne. Posodobitve se najpogosteje izvaja zaradi varnostnih groženj. Ko proizvajalec programske opreme ne zmore več zagotavljati varnosti svojih strank, mora izdelek umakniti s trga.

V družbi Adobe so pravzaprav pred kratkim, in sicer decembra 2020, prenehali podpirati program Adobe Flash, 12. januarja 2021 pa so vsebinam Flash onemogočili delovanje v programu Flash Player[[4]](#footnote-4). Vsem uporabnikom so naročili, naj s svojih naprav odstranijo program Flash Player, da se zaščitijo pred varnostnimi tveganji. To je na žalost zelo slabo vplivalo na spletne založnike in umetnike. Ti so s to programsko opremo namreč ustvarjali svojo digitalno umetnost in spletnemu občinstvu omogočali ogled dinamičnih vsebin in animacij Flash prek spletnega brskalnika. Programska oprema je torej ključnega pomena, pomanjkanje delujoče programske opreme pa neposredno vpliva na to, katere digitalne vsebine lahko še naprej uporabljamo in kakšno znanje lahko prenesemo na prihodnje generacije.

* 1. Zgodovina programske opreme in hitrost staranja

Obstaja veliko različnih vrst programske opreme in mnoge izmed njih si lahko ogledate v spletnem arhivu The Internet Archive Software Collection[[5]](#footnote-5). Zbirka obsega več kot 862.000 paketov programja, od operacijskih sistemov, programov za medijsko produkcijo in statističnih paketov do iger in specializirane programske opreme za 3D-vizualizacijo, zemljevide in animacije. Računalniški muzej v Ljubljani[[6]](#footnote-6) in podobne institucije po svetu[[7]](#footnote-7) rekonstruirajo, ohranjajo in prikazujejo starejšo programsko opremo ter nas tako popeljejo na razburljivo popotovanje skozi razvoj računalniških tehnologij. Treba pa je dodati, da številna programska oprema izven muzeja ni več v uporabi.

*Zastarelost programske opreme je vsesplošen pojav. Vpliva na vso programsko opremo. Gre za naravno posledico inovacij: ko ustvarjamo nove različice programske opreme, stare niso več podprte, varne in uporabne.*

Pravzaprav, hitreje ko uvajamo inovacije, hitreje zastara programska oprema in vedno več naših digitalnih sredstev postane nedostopnih in neuporabnih brez združljive programske opreme. Zastarelost programske opreme še zlasti negativno vpliva na digitalne vsebine, ki imajo dolgotrajno vrednost, kot so viri znanja in digitalna kulturna dediščina. Znanstvenih poskusov na primer ni mogoče zanesljivo ponoviti brez izvorne programske opreme. Vrednost digitalne umetnosti se ne more zviševati, če je ni mogoče pokazati in si je ogledati. Zato je izjemno pomembno, da se programje začne uporabljati dolgotrajno. Na srečo obstajajo načini, ki omogočajo uporabo preteklih podatkov, rekonstrukcijo preteklih raziskav in predstavljanje digitalne umetnosti izpred desetletij.

* 1. Knjižnica programske opreme za podedovano programsko opremo

V družbi Intact Digital[[8]](#footnote-8) smo leta 2016 združili moči v prizadevanjih, da bi ustvarili tehnično učinkovite in ekonomsko vzdržne storitve, ki bi omogočale dolgotrajno gostovanje in vzdrževanje programske opreme, potrebne za digitalno kontinuiteto.

Ker je programska oprema sestavljena iz izvorne kode in izvedljivih datotek, ki jih je mogoče namestiti, se lahko sprejme različne strategije za zagotavljanje nadaljnjega delovanja programske opreme. Izvorna koda odprtokodne programske opreme je običajno dostopna in načeloma ni tveganja za zastaranje programske opreme, če skupnost razvijalcev ohranja znanje in veščine za nadaljnji razvoj programske opreme in lahko zagotovi, da se bodo stari podatki uporabljali z novimi različicami programske opreme.

Vendar pa je velik del programske opreme in našega digitalnega gospodarstva osnovan na lastniški programski opremi, ki je narejena po meri, njena izvorna koda pa ni javno dostopna. Nove različice programske opreme morda niso združljive s prejšnjimi, spreminjanje in ponovni razvoj pa je drag ali neizvedljiv zaradi pomanjkanja dokumentacije in znanja. In kar je še pomembneje, če proizvajalec programske opreme preneha s poslovanjem, programska oprema postane povsem nedostopna. Uporabniki ostanejo brez nadgradenj in sčasoma svojih podatkov ne morejo več uporabljati.

*Bolj ko je programska oprema uspešna, večjo škodo povzroči njena zastarelost.*

V nekaterih primerih obstajajo sodobna programska oprema, ki lahko igra vlogo nadomestka, in obdeluje podatkovne datoteke zastarelih programskih izdelkov. V nasprotnem primeru pa je najboljše ustvariti računalniško okolje, v katerem lahko stara programska oprema še vedno deluje brez varnostnih groženj, kar lahko dosežemo z namestitvijo programske opreme v navidezne naprave (*virtual machine*).

Da bi vzpostavili neko načelo, kako pristopiti k odvisnosti od podedovane programske opreme, smo v družbi Intact Digital zasnovali ogrodje Executable Archive[[9]](#footnote-9), ki tradicionalne arhive dopolnjuje s platformo Software Library in storitvami za zagotavljanje dolgotrajnega gostovanja in vzdrževanja podedovane programske opreme.

Dolgotrajna skrb za programsko opremo pomeni sistematičen pristop k upravljanju z datotekami programske opreme in dokumentacijo, ki je potrebna za nameščanje programske opreme. To vključuje prakse zagotavljanja kakovosti, ki se jih upošteva med namestitvijo programske opreme in stalnim vzdrževanjem navideznih računalniških okolij.

Danes se programsko opremo v navidezna okolja pogosto namešča v podatkovnih centrih in javnih oblakih, ker to omogoča prilagodljivo upravljanje računalniških virov, ki jih potrebujejo organizacije in posamezniki. Mnogi končni uporabniki navidezne naprave vzpostavijo tudi na svojih osebnih računalnikih. Uporabniki Applovih računalnikov Mac lahko na svojih prenosnikih na primer vzpostavijo navidezno okolje Windows.

Navidezne naprave si lahko predstavljate kot namizne računalnike brez lastne strojne opreme, na katere se lahko namesti star operacijski sistem in stare aplikacije. Tako navidezno napravo je mogoče vzpostaviti na sodobnem računalniku, vsa navodila iz aplikacij nameščene programske opreme pa so prevedene v ukaze na računalniku gostitelju. Uporabniki lahko programsko opremo tako uporabljajo na praktično enak način kot v preteklosti, saj se prikaže kot aplikacija v navidezni napravi, ki je uporabniku poznana.

*Slika 1. Ogrodje Executive Archive dopolnjuje tradicionalne prakse elektronskega arhiviranja s postopki in tehničnimi komponentami, ki zagotavljajo dolgotrajno uporabo programske opreme. Platforma Software Library pa omogoča gostovanje in shranjevanje programske opreme ter uporabo programske opreme za obdelavo arhiviranih podatkov na daljavo.*

* 1. Rekonstrukcija analiz podatkov

V strogo reguliranih sektorjih, kot so farmacija in biološke znanosti, je treba digitalne podatke hraniti desetletja. V zapletenih znanstvenih protokolih običajno uporabljajo napredne instrumente za zbiranje podatkov in visoko specializirano programsko opremo, s katero te podatke interpretirajo in analizirajo. Datoteke z neobdelanimi podatki instrumentov se shranjujejo v elektronskih arhivih in redno se preverja celovitost podatkov, npr. z uporabo kontrolnih vsot. Ustrezna programska oprema pa mora ostati funkcionalna, da lahko omogoči rekonstrukcijo preteklih študij in ponovljivost analiz podatkov. To predstavlja velik izziv, ker se tako instrumente kot programsko opremo po določenem času preneha uporabljati.

Platformo INTACT Software Library se zato uporablja za navidezno nameščanje programske opreme, ki je potrebna za rekonstrukcijo preteklih študij. Pri vsem tem je ključno, da nameščena programska oprema v navideznih okoljih zagotovi enake rezultate kot programska oprema, ki je bila prvotno uporabljena v laboratoriju, kjer so se študije izvajale. Poleg tega je treba nadzorovati dostop do virtualizirane programske opreme. Uporabniki morajo namreč upoštevati licenčno pogodbo prodajalca programske opreme in zagotoviti podrobno revizijsko sled njene uporabe, ki jo zahtevajo notranje politike in predpisi.

Z vidika uporabnika je dostop do navideznih naprav na platformi Software Library zelo enostaven prek navideznega namizja v katerem koli novejšem brskalniku (*Slika 2*), kar dosežemo s skrbno izolacijo navideznih naprav, ki gostijo stare operacijske sisteme in zato ne smejo povezane na internet. Kopija podatkov iz elektronskega arhiva pa se varno prenese na Software Library prek namizja Transfer Desktop.

*Slika 2. Portal Software Library lahko uporabljate prek katerega koli novejšega brskalnika. Prek posebej konfiguriranega namizja Transfer Desktop lahko uporabnik prenese kopijo podatkov v Software Library in jo nato obdela z določeno programsko opremo. Vsaka navidezna naprava s programsko opremo se v zavihku brskalnika prikaže kot ločeno namizje.*

Ko so podatki preneseni v okolje Software Library, so uporabljeni v navidezni napravi, ki gosti programsko opremo za obdelavo podatkov, vsaka virtualna naprava pa se nahaja v svojem zavihku brskalnika. Na sliki 3 je prikazan primer navidezne naprave, ki gosti operacijski sistem Windows XP in programsko opremo Analyst 1.4.2 družbe Sciex. Uporabnik lahko programsko opremo in podatke uporablja na enak način kot prvotno v laboratoriju.

*Slika 3. Programska oprema Analyst 1.4.2 družbe Sciex je nameščena na virtualni napravi, ki poganja sistem Windows XP. Do nje se dostopa prek navideznega namizja na katerem koli združljivem brskalniku. Virtualizirano namizje Windows XP se prikaže v ločenem zavihku brskalnika, vendar spletne storitve do njega ne morejo dostopati.*

* 1. Rekonstrukcija digitalne umetnosti

V zadnjih desetletjih so umetniki s spletnimi tehnologijami ustvarjali spletno umetnost in tako dosegli široko spletno občinstvo. Na njihovo ustvarjalnost je zlasti vplival Adobe Flash, ki je bil prej poznan kot Macromedia Flash in FutureSplash. Flash je omogočal prilagodljivo uporabo besedil, vektorske grafike ter uporabo video in zvočnih vsebin za produkcijo animacij, igric, obogatenih spletnih in namiznih aplikacij ter v brskalnik vdelanih video predvajalnikov. Končni uporabniki pa so si lahko vsebino Flash priročno ogledali prek predvajalnika Flash Player na spletnih brskalnikih.

Poleg tega so lahko avtorji po zaslugi standardizacije in prevzetja jezika Virtual Reality Modeling Language (VRML)[[10]](#footnote-10) določili 3D »svetove« in objekte z obogatenimi strukturami, teksturami in interaktivnimi modeli, ki so bili neodvisni od platform. Pregledovalnik Cortona3D[[11]](#footnote-11) za VMRL je omogočal tudi nestandardno podporo za kombiniranje VRML s teksturami Flash, ki so dodale še več možnosti za ustvarjanje.

Nedavno zastaranje programov Adobe Flash in Flash Player je na žalost prizadelo vso digitalno umetnost, ki je narejena z uporabo Flash, zato je potrebno usklajeno prizadevanje, da bo ta ostala dostopna in uporabna.

Intact Digital je s sodobnim umetnikom Michaelom Takeom Magruderjem[[12]](#footnote-12) sodelovala pri rekonstrukciji dela *World[s]* (2006 (v1.0), 2009 (v1.1))[[13]](#footnote-13), ki temelji na VRML in vtičnikih Flash, da bi omogočili 3D-teksturiranje elementov avdiovizualne umetnosti. [Michael Takeo Magruder](http://www.takeo.org) vzdržuje spletni portal, kjer so podrobno opisana njegova umetniška dela, vključno z dokumentacijo, videoposnetki in fotografijami, poleg tega pa upravlja arhiv digitalnih medijskih datotek in izbranih različic programske opreme, s katero so bila umetniška dela ustvarjena in objavljena.

V sodelovanju s tehničnim osebjem družbe Intact Digital smo ustvarili umetniško instalacijo v okolju Software Library, ki ga lahko uporablja spletno občinstvo (sliki 4 in 5).

*Slika 4. Portal Software Library gosti umetniško instalacijo World[s] umetnika Michaela Takea Magruderja, ki vključuje vtičnike Flash in Cortona3D za brskalnike v varnem okolju Software Library.*

*Slika 5. Cortona3D v.7.0 z brskalnikom Internet Explorer 11.1790.17763.0 in vtičnikom Macromedia Flash ActiveX 8r42 je nameščena v izoliranem navideznem stroju z grafično procesno enoto, zato je dostop do filma World[s] zanesljiv in varen s katerega koli sodobnega brskalnika.*

* 1. Dolgotrajna skrb za programsko opremo

Čeprav so tehnični vidiki zastaranja programske opreme očitno težavni, je za zagotavljanje digitalne kontinuitete potreben celosten pristop z upoštevanjem različnih pomislekov. Za namene dolgotrajnega vzdrževanja programske opreme na portalu Software Library smo razvili prakse zagotavljanja kakovosti, ki zajemajo tehnološke, pravne, operativne in človeške dejavnike (*Slika 6*).

*Slika 6. Pri dolgoročni skrbi za programsko opremo je treba upoštevati več dejavnikov in postopkov za zagotavljanje kakovosti, s katerimi lahko prepoznamo in zmanjšamo morebitna tveganja.*

Kar zadeva tehnološke dejavnike, pokrivamo varnost in integriteto dolgotrajne hrambe datotek in dokumentacije programske opreme, računalniška okolja, ki so potrebna za nameščanje in delovanje programske opreme, ter metode za varen dostop do nameščene programske opreme in njeno uporabo.

Med pravne vidike na primer spadajo licenciranje vseh tehničnih komponent, ki so vključene v nameščanje in uporabo programske opreme. Operativne dejavnosti pa vključujejo stalno vzdrževanje računalniškega okolja ter zaščito in morebitno ponovno namestitev programske opreme, da se podaljša njeno uporabo. Čeprav človeške dejavnike pogosto spregledamo, so bistvenega pomena. Treba je zagotoviti usposabljanje za nameščanje podedovane programske opreme in ohranjanje znanja, da bodo lahko mlajše generacije zanesljivo in učinkovito uporabljale zgodovinsko programsko opremo in zgodovinske podatke. Varno nameščanje je še posebno pomembno, ker stara programska oprema v osnovi ni varna, če je izpostavljena sodobnemu ekosistemu. Brez nadgradenj morda ne bo delovala na najnovejših operacijskih sistemih, poleg vsega pa je dovzetna za kibernetske napade.

Na splošno lahko s sistematičnim in vodenim pristopom, ki ga ponuja okvir Executable Archive, ter priročno uporabo programske opreme, ki gostuje na portalu Software Library, učinkovito obravnavamo raznovrstne primere zastaranja digitalnih vsebin. Na tej osnovi lahko znatno prispevamo k digitalni kontinuiteti in nadaljujemo inoviranjem, saj vemo, da je naša digitalna dediščina varna in bo dostopna tako dolgo, kot bo to potrebno.

1. 5. Partner – Večno mlad

Boštjan Špetič (Računalniški muzej, vodja kustodiata)

Računalniški muzej v Ljubljani (zasebni muzej, ki je bil ustanovljen leta 2004) zbira strojno in programsko opremo posebnega pomena za slovensko družbo. V skladu z zbiralno politiko Računalniškega muzeja[[14]](#footnote-14) imajo predmeti, ki so izdelani v Sloveniji ali so delo slovenskih izumiteljev, glede na merila zbiranja najvišjo prioriteto. Računalniški muzej hrani verjetno največjo zbirko računalnikov, izdelanih v Sloveniji. Morda celo bolj kot na oddelkih za konservacijo in restavriranje, si v okviru delovanja muzejskega eksperimentalnega laboratorija aktivno prizadevamo, da bi vse naprave ponovno začele delovati. Gre za zahteven podvig, pri katerem so potrebni ločeni protokoli in sodelovanje s strokovnjaki z različnih področij,[[15]](#footnote-15) vključiti pa je treba tudi digitalno arheologijo in rekonstrukcijska prizadevanja. Ta muzejski proces zato nudi edinstven vpogled v izzive in priložnosti, ki jih prinaša ohranjanje programske opreme.

Med rutinskim varnostnim kopiranjem trdega diska enega izmed slovenskih računalnikov smo našli pravo zakladnico izvornega programja. Programje je na splošno najtežje najdljiv in ohranljiv del računalniške zgodovine, saj večina računalnikov, s katerimi delajo muzeji, nima trdih diskov, pa tudi zato, ker generacije pred nami niso ravno dobro tehnično in sistematično ohranile disket in drugih pomnilnih enot.

V tem članku bo avtor opisal pomen odkritja iz trdega diska in prikazal, kako zapleteno je bilo arhiviranje softverske dediščine.

* 1. Računalnik Iskra Delta Partner

V letu po sloviti združitvi družb Iskra in Delta[[16]](#footnote-16) je šest strokovnjakov nove družbe v manj kot dveh mesecih razvilo prototip namiznega mikroračunalnika, ki so ga poimenovali Partner. Namenjen je bil predvsem uporabi v manjših in novoustanovnih organizacijah, tržili pa so ga tudi kot sistem, ki je primeren za raziskovalne dejavnosti, šole in celo posameznike.

*Slika 1: Računalnik Partner iz zbirke Računališkega muzeja. (foto: arhiv Računalniškega muzeja)*

Delta Partner je bil namizni računalnik, zasnovan kot praktičen poslovni ali razvojni sistem, ki se ga je uporabljalo predvsem v računovodstvu, knjigovodstvu in za statistične obdelave. Računalnik Partner je bilo mogoče vključiti v omrežja z večjimi računalniki, kjer je deloval kot terminal in tudi kot samostojen sistem. Po letu 1983 so izdelali več modelov z enakimi osnovnimi lastnostmi in poljubnimi razširitvami za vsak model. Sistem je bil zgrajen na osnovi že takrat nekoliko zastarelega 8-bitnega mikroprocesorja Zilog Z80A s frekvenco 4 MHz, običajno s 128 kB pomnilnika RAM in 8 kB pomnilnika ROM.[[17]](#footnote-17)

Računalnik so prodajali v paketu z namensko programsko opremo, ki je na začetku zajemala le štiri uporabniške programe in operacijski sistemi CP/M 3. Štirje uporabniški programi MIPOS so bili namenjeni obdelavi posameznih ali povezanih poslovnih podatkov, ki so omogočali knjigovodstvo prek glavne knjige, vodenje knjige terjatev kupcev in dobaviteljev, pa tudi vodenje skladiščne logistike in fakturiranje. Programska orodja so sčasoma omogočila tudi opravljanje drugih pisarniških opravil, kot je obdelava dokumentov. Kasneje pa so izdali še več orodij, ki so podpirala izvajanje znanstvenih izračunov na področju upravljanja energije, strojništva in gradbeništva, ter orodij za procesno podporo v živilskopredelovalni in farmacevtski industriji. S tem so dopolnili ponudbo programske opreme za interaktivno programiranje, klasičnih prevajalnikov za programske jezike Cobol, Basic, Pascal in C ter makrozbirnik za simbolni strojni jezik. Prve različice sistema so prikazovale le besedilo, novejše pa so bile nadgrajene z grafično kartico z ločljivostjo 1024 × 512 točk, zato je bilo podatke mogoče prikazati v grafični obliki s pomočjo programskih paketov BGRAF, IDRIS in DIAS. Med prvimi tremi modeli je bil WF/G prvi, ki je imel vgrajeno disketno enoto in disk Winchester z 10 Mb, modela 1F/G in 2F/G pa sta imela po eno ali dve disketni enoti. Vsi računalniki Partner so imeli serijska vrata za povezovanje s tiskalnikom in miško, omogočali pa so tudi namestitev dodatnih vrat. Običajno je sistem vključeval tudi tiskalnik TRS835 in možne druge razširitve, npr. omrežna vrata in TV-izhod.[[18]](#footnote-18)

Proizvodnja je sprva potekala v Kranju, pozneje pa so jo preselili na Ptuj. Proizvodnjo za zahodne trge so postopoma vzpostavili v novi tovarni družbe Delta v Avstriji, v družbi Novkabel v Novem Sadu pa so izdelovali poseben model računalnika Partner z drugim paketom računalniške programske opreme. V avgustu leta 1983 je bilo na testiranje pripravljenih prvih deset sistemov iz redne proizvodne linije, do konca leta pa so jih v Kranju sestavili že 170. V avstrijsko tovarno St. Jakob so poslali 200 enot, kjer naj bi do konca leta sestavili vsaj 100 Partnerjev za izvoz. Od vseh slovenskih računalnikov je bil to daleč najbolj razširjeni model, saj so skupno proizvedli več tisoč računalnikov Partner, načrtovali pa so proizvodnjo vsaj 10.000 sistemov. Leta 1988 je bilo v registru računalniške opreme[[19]](#footnote-19) zabeleženih približno 450 računalnikov Partner, ki so se še vedno uporabljali v Sloveniji. Danes vemo, da obstaja še 12 tovrstnih naprav, predvidevamo pa, da jih največ dvakrat ali trikrat toliko hranijo zasebni zbiratelji ali pa so kje založene in samo čakajo, da jih nekdo odkrije.

Računalnike Partner so imeli na voljo le v peščici slovenskih šol, med prvimi so jih dobili v Srednji tehniški šoli Kranj. Tam je računalniški predmet v poznih 80. letih prejšnjega stoletja vodil Marko Grobelnik, ki je pozneje postal vodja raziskovalnega oddelka za strojno učenje na Institutu Jožef Stefan.[[20]](#footnote-20)

* 1. Posnemanje računalniškega okolja Partner

Zaradi propada družbe Iskra Delta, razpada Jugoslavije, prehoda v kapitalizem in napredka arhitekture IBM PC je računalnik Partner zastaral in nanj so kmalu praktično povsem pozabili.

Leta 2018 je slovenski študent računalništva z metodo obratnega inženirstva raziskal notranjost računalnika in napisal posnemovalnik (emulator), ki je prvič po 25 letih omogočil ponovni zagon izvirne programske opreme.[[21]](#footnote-21) Ta študent, Matej Horvat, je danes del ekipe Računalniškega muzeja in kot strokovnjak posebno pozornost namenja računalnikom Partner.

Takrat se je majhna skupnost »Part-Time Nerds«[[22]](#footnote-22) lotila izziva, da s tem emulatorjem spiše novo programje za računalnike Partner. Horvatov emulator deluje tudi v spletnem okolju, zato smo ga lahko namestili na spletno stran Računalniškega muzeja, da ga lahko preizkusijo vsi, ki jih to zanima.[[23]](#footnote-23)

* 1. Programska oprema Partner je prava redkost in zgodovinska zanimivost

Emulator je bil napisan na računalniku Partner, ki so ga skrbno hranili na Fakulteti za računalništvo in informatiko Univerze v Ljubljani. Kolikor vemo, sta njihov in naš računalnik edina Partnerja z delujočim trdim diskom, zato sta tudi edina vira izvirne programske opreme, ki se je ohranila vse do danes.

Celoten seznam trenutno ohranjenega programja obsega programske jezike Microsoft, nekaj izvirne programske opreme za storilnost Iskra Delta, delno lokaliziran urejevalnik besedil Wordstar in številne aplikacije, ki so jih razvili uporabniki.

Ena najzanimivejših najdb izvornega razvoja, ki smo jo odkrili znotraj medijske arheologije pri Računalniškem muzeju je različica igre kača[[24]](#footnote-24) z imenom Glista[[25]](#footnote-25). Zanjo je bila napisana osupljivo obsežna dokumentacija. Tukaj je prikazan prevod pozdravnega zaslona, kjer je jasno opisan namen avtorja, ki povzema določena človeška prizadevanja tistega časa:

»Ta igra je bila ustvarjena kot sredstvo za premagovanje dolgčasa, ki se navadno pojavi med poukom (tudi med odmorom). Od sebe ne daje nobenih zvočnih efektov, razen ropotanja disketne enote. Če izključiš še KLIK, boš povsem neslišen (preverjeno). Igra ne zahteva večjih umskih naporov, tako da lahko z levo polovico možganov slediš profesorjevi razlagi. Če se ti bo sprva zdelo, da ne boš nikoli dosegel spodobnega števila točk in boš že skoraj obupal nad usodo, vedi, da ti bo pomagal le trening.«

*Slika 2 Zajem pozdravnega zaslona iz igre Glista. (foto: arhiv Računalniškega muzeja)*

*Slika 3 Zajem naslovnega zaslona iz igre Glista. (foto: arhiv Računalniškega muzeja)*

Gre res »samo« za igro, ki je v bistvu kopija druge dobro znane igre, vendar smo se vanjo poglobili prav zaradi podrobnosti in specifičnosti pričakovane uporabe.

* 1. Arhiviranje programja kot temeljni muzejski postopek

V tem poglavju bo avtor na kratko opisal, kako v Računalniškem muzeju razumemo arhiviranje programja. Kot smo to predstavili na letni konferenci NetPreserve,[[26]](#footnote-26) imamo muzejski strokovnjaki nalogo, da ohranimo čim več gradiva za prihodnje generacije raziskovalcev. Po eni strani poskušamo ohraniti čim več »informacij«, po drugi pa opazujemo avro fizičnih predmetov. Naša naloga pa je vzdrževati znanje, ki je potrebno za tehnično razumevanje predmetov, in tudi komuniciranje z javnostjo. Vse to počnemo, ko uspemo izvesti živo rekonstrukcijo koščka programske kode.

Za ilustracijo različnih stopenj lahko uporabimo model oz. tehnično skico za programje s tekstovnim vmesnikom, ki bi se verjetno izvajal na računalniku Partner.

*Slika 4 Tehnična skica za program, ki bi se verjetno izvajal na računalniku Partner. (foto: arhiv Geodetski vestnik)*

Druga skrajnost je popolna produkcijska omara največje slovenske elektronske oglasne deske (BBS, Bulletin Board System). Ekipa Računalniškega muzeja je omaro odpeljala neposredno iz sobe, kjer je nazadnje delovala in služila uporabnikom. Trenutno poskušamo obnoviti disk. To je primer programja, ki je bilo standarndno dostopno, vendar so komunikacija in podatki, ki so jih uporabniki pustili za sabo na strežniku neprecenljivi.

*Slika 5 Produkcijska omara največje slovenske elektronske oglasne deske (BBS). (foto: arhiv Računalniškega muzeja)*

Pri postopku ohranjanju programja, sledimo naslednjim korakom:

1. Dokazi o njegovem obstoju. Omembe v intervjujih ali sočasnih člankih.
2. Dokumentirani vhodni in izhodni podatki. Posnetki zaslona ali priročniki.
3. Binarne datoteke. Pogosto so rešene z diskov in trakov.
4. Binarne datoteke + podatki. Pridobljeno iz nekakšnega produkcijskega okolja.
5. Izvorna koda.
6. Bonus: izvirna strojna oprema.
7. Dodaten bonus: dejanska izvorna naprava, ki je poganjala to specifično programje.

Najvišji cilj po katerem koli koraku pa je živa rekonstrukcija. Pri vsaki višji stopnji ohranjenosti je rekonstrukcija lažja, verjetnejša in bolj celostna. V praksi vsakič občutimo neznansko srečo, ko uspemo priti vsaj do (3) ravni binarne datoteke, torej da uspemo doseči nekaj več kot le dokazati obstoj.

Vsi programi, ki smo jih našli na edinih dveh računalnikih Partner s trdim diskom, so bili zapisani v binarni obliki, zato smo jih lahko zagnali na izvirni strojni opremi ali v emulatorju. Omogočajo nam vpogled v nekdanjo uporabo programske opreme, ne pa tudi v način razmišljanja avtorjev.

Program Glista je med njimi izjema, saj je bilo zelo veliko pozornosti posvečene podrobnostim, na voljo je bila celotna dokumentacija in pozdravni zaslon z zgoraj prikazanim motivacijskim sporočilom, poleg vsega pa je bil program tudi podpisan.

* 1. Neločljivi entiteti uporabnika in naprave

Če je le mogoče, se na našem raziskovalnem oddelku radi opravimo pogovore z uporabniki in avtorji, da ustvarimo zapis z izkušnjami iz prve roke. Priložnosti za to pa je izjemno malo.

Poiskali smo avtorja programa Glista, ki je danes visoki uradnik na Ministrstvu za javno upravo. Obiskal je muzej in pokazali smo mu delujoč računalnik Partner, ki ga je pred 35 leti uporabljal na Srednji tehniški šoli Kranj pod vodstvom profesorja Grobelnika.

G. Gabrijel je zelo resen človek in njegov smisel za podrobnosti pri programu Glista kaže, da je tudi zabavo v srednješolskih letih jemal zelo resno. Iz srca se nam je zahvalil za ponovno in ganljivo srečanje s svojim programom.

* 1. Zaključek

To lepo prikazuje človeški vidik pri ohranjanju programja in tudi njegovo sposobnost spodbujanja čustvenega odziva. Razvijalec kanček sebe v digitalni obliki pusti v izvajajoči se kodi programja, tako kot umetnik del sebe pusti v umetniškem delu.

Softverska dediščina je dediščina človeške bistroumnosti. Zajema duh *homo sapiensa*, kot ga poznamo v zadnjih stoletjih, in priča o vzponu morebitnega prihodnjega *homo digitalicusa*.

* 1. Pripis

Interna zasnova računalnika Partner ne omogoča prikaza datumov po letu 1999, zato se je Horvat odločil to popraviti v posnemovalniku s ponastavitvijo katerega koli datuma iz 21. stoletja nazaj na osnovno leto 1990.

Programi v emulatorju tako tečejo v večnih 90. letih, ki predstavljajo zlato dobo, ki je računalniki Partner nikoli niso doživeli, ker so osebni računalniki prevladali na tržišču in ker je družba Delta kot prva v samostojni Sloveniji leta 1990 razglasila stečaj, ki se je končal šele leta 2020. Stečajni upravitelj je žal že na začetku postopka uničil vso tehnično dokumentacijo.

Glista pa bo tako večno živela v našem Partnerjevem emulatorju in »žurala«, kot da so še vedno devetdeseta.

1. 6. Pravni izzivi arhiviranja programske opreme

Matija Šuklje

V tem prispevku bomo na kratko obravnavali glavne pravne izzive, ki se pojavljajo pri arhiviranju programske opreme v muzeju. Ker so avtorske pravice glavne pravice »intelektualne lastnine«, ki so odgovorne za zaščito in omejevanje uporabe programske opreme, bomo situacijo analizirali predvsem skozi to prizmo. Kljub temu pa bomo, kjer je to relevantno, omenili tudi druge pravice.

Dejstva bomo poskušali predstaviti kar se da splošno, ker pa se Računalniški muzej in avtor prispevka nahajata v Sloveniji, se bomo osredotočili na to jurisdikcijo.

* 1. Pravice »intelektualne lastnine«

Izraz pravice »intelektualne lastnine«[[27]](#footnote-27)(»PIL«) razumemo kot izključne pravice na neopredmetenih sredstvih.

Pomembno je vedeti, da izvirni ustvarjalec – kot je avtor na področju avtorskih pravic ali izumitelj na področju patentov – morda ni dejanski imetnik izključnih pravic. Svoje pravice lahko ustvarjalci prenesejo na nekoga drugega s pogodbo ali pa zato, ker so denimo stvaritev ustvarili znotraj delovnega razmerja s svojim delodajalcem.

Razen če vam zakon ali imetnik pravic ne podeli posebnih pravic, ki se nanašajo na stvaritev, so pravice izključno v rokah njihovega imetnika.

V naslednjih podpoglavjih bomo obravnavali vsako pravico intelektualne lastnine, ki igra pomembno vlogo pri arhiviranju programske opreme, in sicer v padajočem vrstnem redu glede na pomembnost oz. tveganje.

* + 1. Avtorske pravice

Avtorske pravice (ang. copyright) so glavne pravice intelektualne lastnine, ki se nanašajo na programsko opremo.

Zajemajo stvaritve (in ne »zgolj idej«) in samodejno nastanejo v trenutku, ko je izvirno umetniško delo izraženo oz. fiksirano v mediju. Zaščitene so po celem svetu.

Test izvirnosti je pomemben, vendar je nekoliko nenavaden. Po eni strani omogoča obstoj dveh povsem enakih del, ki sta zaščiteni z avtorskimi pravicami, če sta oba avtorja svoji deli ustvarila popolnoma neodvisno drug od drugega in so podobnosti zgolj naključne. Po drugi strani pa test izvirnosti preprečuje, da bi bilo mogoče avtorsko zaščititi trivialne zadeve, kot so dejstva, postavitvene datoteke itd.

V večini jurisdikcij avtorske pravice trajajo za čas življenja avtorja in 70 let po njegovi smrti. Če je imetnik avtorskih pravic pravna oseba, pa veljajo 70 let od stvaritve. Po tem obdobju delo postane javno dostopno in z njim lahko vsakdo počne, kar želi.

* + 1. Patenti

Patenti, vsaj v Evropski uniji, »za programsko opremo ne obstajajo *per se*.« V praksi pa se je temu mogoče izogniti na več načinov, saj je to poslovni interes nekaterih (vendar ne vseh) IT-podjetij in patentnih uradov. Zato moramo obravnavati tudi patente.

V začetnem obdobju razvoja računalniške programske opreme je dejansko prišlo do živahne razprave o tem, ali naj se programsko opremo zaščiti s patenti ali avtorskimi pravicami. Zmagale so avtorske pravice, vendar je na koncu programska oprema postala edino ustvarjalno delo, ki je lahko hkrati zaščiteno z avtorskimi pravicami in patenti.

Patent se nanaša na izum, ki mora biti nov, imeti inventivni korak in biti industrijsko uporaben – abstraktne ideje in matematični algoritmi namreč niso dovolj. V praksi je riguroznost preverjanje teh testov odvisna od jurisdikcije in patentnega urada, kar pomeni, da bi lahko mnogi (softverski) patenti[[28]](#footnote-28)postali neveljavni, če bi jih kdo poskušal razveljaviti s tožbo.

Patenti morajo biti registrirani v vsaki državi, v kateri jo izumitelj poskuša zaščititi, in zaščita običajno traja 20 let. Po tem obdobju patenti potečejo in postanejo javni.

* + 1. Blagovne znamke in storitvene znamke

Blagovne znamke ščitijo izvor/znamko blaga, storitvene znamke[[29]](#footnote-29) paizvor/znamko storitev. Da lahko opravljajo svojo vlogo in so veljavne,[[30]](#footnote-30) pa morajo imeti razlikovalni učinek in jih je potrebno uveljavljati.

Običajno se blagovno znamko (®) registrira za obdobje 10 let, kar se lahko po želji podaljša. Registracija je omejena na državo in na razrede blaga/storitev, za katere ste vložili prijavo.

Obstaja tudi koncept »slovečih blagovnih znamk« – če je znamka dobro poznana po celem svetu, lahko lastnik drugim osebam prepreči uporabo blagovne znamke tudi za tiste razrede, za katere lastnik ni vložil prijave (npr. znamka tipkovnic Nutella).

Poleg tega zakon do neke mere varuje tudi neregistrirane blagovne znamke (™), dokler »imetnik« takšne znamke uveljavlja to blagovno znamko.

Na področju programske opreme lahko na blagovne znamke naletite pri imenih in logotipih družb, projektov, izdelkov in storitev.

* + 1. Podatki v programski opremi/s programsko opremo

Programska oprema je pogosto uporabna in zanimiva le zaradi podatkov, ki jih obdeluje. Ko razpolagamo s podatki, je treba upoštevati naslednje razlikovanje.

#### Podatki

Dejstva, ki so kot taka goli podatki, niso zaščitena z avtorskimi pravicami. Kljub temu pa je del podatkov mogoče zaščititi z avtorskimi pravicami (npr. leposlovna besedila, slike, avdio- in videoposnetke itd.), z blagovnimi znamkami (imena znamk, logotipi) itd.

Če obdelujete osebne podatke, je treba nadalje upoštevati zakonodajo o varstvu osebnih podatkov, kot je GDPR.

#### Baza podatkov

Podatke se pogosto zbira in ureja v bazi podatkov. Baze podatkov *kot celota* so pogosto zaščitene s pravicami na področju baze podatkov, kot je direktiva o pravnem varstvu baz podatkov v EU,[[31]](#footnote-31)medtem ko lahko (nekatere ožje) avtorske pravice še vedno veljajo za prvotno ureditev, izbiro in predstavitev podatkov.

Pravice baze podatkov v skladu z direktivo o pravnem varstvu baz podatkov veljajo v obdobju 15 let in zanje velja samodejno učinkovanje. Z vsako bistveno spremembo baze podatkov pa se za to bazo podatkov oblikuje nov sklop pravic. Lastnik ima pravico do ugovora zoper kopiranje bistvenih delov svoje baze podatkov, tudi če se podatki pridobivajo in rekonstruirajo postopno.

* + 1. Oblikovanje uporabniškega vmesnika (UI)/uporabniške izkušnje (UX) programske opreme

Računalniški program ima običajno (grafični) uporabniški vmesnik. Naslednje pravice se lahko nanašajo na splošni »videz in občutek« programske opreme ali spletne strani:

* *Avtorska pravica* – šibka in ozko usmerjena (zaradi dihotomije ideja–izraz), vseeno pa so lahko z njo zaščitene ikone ter avdio- in videoposnetki, če so dovolj izvirni.
* angl. *»Trade dress« (*nem. *Geschmacksmuster; nimamo slovenskega izraza*) – poseben videz in občutek, ki mora biti razločevalen in nefunkcionalen, ni ga treba registrirati, vendar je temu težko zadostiti.
* *Pravice iz modela* oz. angl. *industrial design rights* (npr. modeli v ZDA, angl. *design patents*) – treba jih je registrirati (npr. vojna med družbama Apple in Samsung zaradi podobnosti med ikonami).

V praksi je malo verjetno, da bi zaradi njih prišlo do kakršnih koli težav pri predstavljanju programja v muzeju/arhivu.

* + 1. Strojna oprema

#### Vezja strojne opreme, sheme itd.

Čeprav ob arhiviranju programske opreme same ni verjetno, da bi naleteli na različne sheme programja, načrte vezij itd., pa so ti verjetno zelo zanimivi za Računalniški muzej.

Takšne elemente je mogoče zaščititi z različnimi pravicami integriranega vezja. V ZDA se pogosto imenujejo »mask works«, v EU pa običajno govorimo o »pravnem varstvu topografije polprevodniških izdelkov«. Te pravice so običajno bolj omejene kot avtorske pravice ali patenti in v več jurisdikcijah so samodejne pravice.

* 1. Imena domen
     1. Imena internetnih domen

Imena internetnih domen ne sodijo med pravice intelektualne lastnine, čeprav je Evropsko sodišče za človekove pravice razsodilo, da gre za »lastnino« (*ESČP: Paeffgen GmbH proti Nemčiji 25379/04*). V praksi se lahko blagovne znamke uporablja za pravno obrambo v sporih glede imen domen.

* 1. Kako pridobiti pravice

Razen če ste dejanski imetnik pravic, morate pravice pridobiti z licenco ali na drugačen način.

* + 1. Brezplačne (oziroma »svobodne«)

Obstajata dva načina, kako lahko programsko opremo vsi uporabljajo brezplačno in na zanesljiv način.

Prvi način je, da so avtorske pravice in morebitni patenti, ki veljajo za programsko opremo – morda celo blagovne znamke, a to ni tako preteče – že potekli. To pomeni, da so na razpolago javnosti[[32]](#footnote-32) in da jih lahko vsi brezplačno uporabljajo.

Druga možnost pa je, da se programsko opremo izda kot brezplačno in odprtokodno. Ker FOSS (Free & Open Source Software) vsem uporabnikom omogoča *uporabo*, *proučevanje*, *deljenje* in *izboljševanje*, je to za potrebe muzeja (veliko) več kot dovolj.

* + 1. Izjeme, določene z zakonom

Še enega od načinov predstavljajo zakonsko določene omejitve in izjeme, kar pomeni, da pravice podeli sam zakon.

*Priobčitev javnosti na zaslonih* (ZASP,[[33]](#footnote-33) 49.b člen) omogoča uporabo na razstavah in to zagotovo velja tudi za omogočanje dostopa (do programja ali drugega gradiva) prek računalniškega zaslona za raziskovanje in učenje na podlagi eksponata.

*Privatno in drugo lastno reproduciranje* (ZASP §50) javnim muzejem, arhivom itd. omogoča reproduciranje na katerem koli nosilcu, če to storijo iz lastnega primerka in če pri tem nimajo namena dosegati gospodarske koristi.

*Prosto reproduciranje in dajanje na voljo javnosti osirotelega dela* (ZASP 50.a) se lahko uporabi, kadar se odkrije osirotelo delo brez avtorja. V takšnih primerih zakon javnim muzejem, arhivom in podobnim institucijam dovoli digitalizacijo, prosto reproduciranje, prikazovanje in celo distribuiranje. Zakon predpisuje, da je treba slediti postopku skrbnega iskanja in beleženja. Velik zaplet predstavlja dejstvo, da v 50. členu programska oprema ni šteta med potencialno osirotela dela, zato bi bilo smiselno predlagati spremembo zakona.

* + 1. Licenca

Ker praktično nobena programska oprema ni v javni domeni in ZASP praktično ne opredeljuje izjem na področju programske opreme, boste skoraj zagotovo morali pridobiti licenco.

Ko se z imetniki pravic pogajate za pridobitev licence, morate razmišljati na široko in vključiti splošno javnost ali pa vsaj druge muzeje/arhive.

Z licenco *FOSS* lahko dejansko najlažje pridobite vse potrebne licence. Manj kot je licenca[[34]](#footnote-34) omejujoča, na manj omejitev boste (in bodo drugi) naleteli ob uporabi pridobljene programske opreme.

Če to ni mogoče, se poskušajte dogovoriti za licenco za *muzejske (in arhivske) namene*. Omejitev licence za primer posebne uporabe bi drugim muzejem (in arhivom) še vedno omogočala uporabo tega dela, poleg tega pa bi odpravila morebitne težave, ki bi doletele vaš muzej v primeru razstavljanja na drugi lokaciji.

Če so pogajanja izjemno težka, obstaja možnost, da se programsko opremo deponira v hrambo (angl. escrow) in/ali podpiše začasno pogodbo o nerazkritju informacij (angl. NDA). Tako lahko zaščitite programsko opremo v svojem muzeju ali arhivu in se z imetnikom pravic dogovorite, da jo prepusti javnosti (v idealnih razmerah kot FOSS) po nekem določenem obdobju ali izpolnitvi določenih pogojev.

* + 1. Javni organi

Če je bila zadevna programska oprema razvita v sklopu javnega organa ali po njegovem naročilu, se lahko prek Zakona o dostopu do informacij javnega značaja (ZDIJZ) zahteva dostop do izvorne kode in pravice do ponovne uporabe.

Po celotni EU je to postalo znano pod načelom *javni denar, javna koda* in zakon v nekaterih državah (npr. v Italiji, Franciji) to tudi izrecno zahteva. V slovenskem zakonu programska oprema sicer ni izrecno omenjena, vendar bi z razumno interpretacijo lahko prišli do takšnega zaključka. To še ni bilo preverjeno v praksi.

* 1. Drugi premisleki

Vse spletne shrambe izvorne kode (kot sta GitHub ali GitLab), galerije slik, spletne strani družbenih omrežij itd. imajo objavljene *Splošne pogoje poslovanja*, ki veljajo tudi, če se nanašajo na muzejsko rabo, in pogosto zajemajo pravice intelektualne lastnine.

Po drugi strani pa obstaja velika verjetnost, da boste sčasoma prejeli pritožbe in zahteve za *izbris* zaradi (zatrjevanih) kršitev avtorskih pravic in odstranitev osebnih podatkov (GDPR). Zato delujte proaktivno in oblikujte politiko, ki bo to urejala.

1. 7. Pionirski začetki uvajanja računalniške podpore pri urbanističnem načrtovanju v obdobju 1970–1985

Franc J. Zakrajšek

1. Uvod

V začetku osemdesetih let 20. stoletja se je v svetu začelo intenzivno uvajanje računalniške tehnologije na različnih strokovnih področjih in sektorjih. Računalniška tehnologija se je po vsem svetu uvajala tako rekoč od samega začetka. To je povzročilo korenite spremembe na strokovnih področjih, vključno z vsebino, delovnimi procesi in uporabo.

V članku delim svoj prispevek k raziskavam, razvoju in implementaciji računalniške podpore na področju urbanističnega načrtovanja v tem obdobju. Pri svojem pionirskem delu na tem področju sem sodeloval z inovativnimi in učinkovitimi interdisciplinarnimi ekipami matematikov, računalničarjev, arhitektov, geografov, ekonomistov, sociologov in drugih strokovnjakov. Sem raziskovalec, razvijalec in svetovalec, zaposlen na Urbanističnem inštitutu Republike Slovenije.[[35]](#footnote-35) Okvir prispevka zagotavlja le skromen prikaz izbranih delcev tistega časa, da bi bralci dobili nekaj vtisov o obravnavanem obdobju ter o naši takratni vlogi in prispevkih.

2. Informacijski sistem za prostorsko načrtovanje

Razvili smo informacijski sistem za prostorsko načrtovanje (urbanistično/regionalno) (ISSP) z dvema komponentama – bazo podatkov in naborom postopkov. Baza podatkov sestoji iz podatkov o preteklih in sedanjih stanjih ter podatkov o odločitvah glede načrtovanja v obliki, primerni za uporabo v postopkih načrtovanja. Bistveni del baze podatkov predstavljajo geokodirani (geografski referenčni) podatki. Nabor postopkov sestavljajo postopki za ustvarjanje in posodabljanje baze podatkov ter za njeno uporabo in analizo, glavni del pa predstavljajo postopki za uporabo podatkov, kot so pridobivanje podatkov, metode statistične analize ter urbanistični in regionalni modeli.

ISSP je sistem za podporo odločanju, namenjen zadovoljevanju podatkovnih potreb znotraj različnih procesov načrtovanja in spremljanja na komunalni, občinski in/ali regionalni ravni. Gre za uporabniško usmerjen informacijski sistem, katerega funkcija ni zbiranje podatkov, pač pa obdelava podatkov za končno uporabo. ISSP ni samostojen informacijski sistem, temveč je tesno povezan z drugimi informacijskimi sistemi v državi. Uporablja namreč podatke iz različnih informacijskih sistemov (npr. administrativne podatke, statistične podatke) ter ustvari podatke, ki bi se lahko uporabljali zunaj območja načrtovanja (npr. podatke za lokalno upravo).

Primarni razvoj v pristopu tega sistema (z vidika inovativnosti in izziva) zajema vključevanje najnovejših spoznanj iz teorije informacijskih sistemov, predvsem tiste, ki jo je promoviral takratni poznavalec na tem področju James Martin. Njegov manifest obravnava informacijske sisteme; trditev, da so podatkovne entitete in s tem podatki bolj stabilni od postopkov, ki jih želimo avtomatizirati, in da so relacijske podatkovne baze v tretji normalni obliki, inženiring informacijske tehnologije, računalniško podprt programski inženiring, programski jeziki četrte generacije, RAD (Rapid Application Development – hiter razvoj aplikacij) in še več. Vse to sem uspešno povezal s svojim matematičnim znanjem o topoloških strukturah (točka, črta, poligon ...) in njihovih medsebojnih odnosih, ki danes predstavljajo osnovo geografskih informacijskih sistemov.

*Slika 1: Predstavitev v fizičnem prostoru, matematika geo-topološkega jedra in tretja normalna oblika. Vir: Zakrajšek, F.: Zasnova informacijskega sistema na ravni mesta Ljubljane. Ljubljana: Urbanistični inštitut Republike Slovenije, 1981*

*Slika 2: Standardna baza podatkov informacijskega sistema za prostorsko načrtovanje. Vir:* Zakrajšek, F. Information systems for urban and regional planning. V: Gerckens, L. C. (ed.), et al. *Planning in transition: regional development, urban form, information system: symposium 1985, Columbus, 21.–22. maj 1985*. Columbus: Urbanistični inštitut Republike Slovenije, Državna univerza Ohia, oddelek za mestno in regionalno načrtovanje, 1985

3. Primer zajema podatkov : centroidi hišnih številk

V 80. letih prejšnjega stoletja je zajem grafičnih podatkov predstavljal poseben izziv. Optični bralniki večjih dimenzij niso obstajali, manjši od A4 pa so bili težko dostopni. Zato smo morali razviti polavtomatske, delno ročne postopke digitalizacije ter celo vrsto programskih vmesnikov za prenos podatkov z luknjanega traka in na luknjan trak. Naj omenim še naš poseben inovativen pristop k digitalizaciji hišnih številk. Koordinate hišnih številk so v urbanih sistemih nujne, saj lahko na ta način spremljamo stanje »bivalne« dejavnosti (stanovalcev in stanovanj) ter dejavnosti »dela« v prostoru. Najprej smo uspešno digitalizirali hišne številke v Celju (približno 20.000 enot) in nekoliko kasneje še hišne številke v Ljubljani. Njihova položajna točnost v naravi je bila 5 m. Geodetska uprava Republike Slovenije je uvedla našo metodo pri digitalizaciji vseh hišnih številk v Republiki Sloveniji. Podatke digitaliziranih hišnih številk v Celju in Ljubljani so prenesli v svojo bazo EHIŠ, ki jo še vedno redno vzdržujejo. Treba je poudariti, da je bila Slovenija med prvimi državami na svetu, ki je digitalizirala vse hišne številke na svojem ozemlju.

*Slika 3: Zajem podatkov, centroidi hišnih številk. Vir: Zakrajšek, F. et al: Informacijska zasnova občine Celje ob pripravi prostorskega plana. Ljubljana: Urbanistični inštitut Republike Slovenije, 1979*

4. Primer izpisa podatkov: prvi digitalni zemljevidi

Podobno kot pri optičnih bralnikih velikega formata je bila v začetku 80. let prejšnjega stoletja težava tudi z risalniki in tiskalniki velikega formata. Da bi bili prvi digitalni zemljevidi čim bolj dostopni, smo sprva uporabljali navadne vrstične tiskalnike in s pretiskom ustvarili manj temnejših področij na zemljevidu. Težavo z velikostjo znakov (pravokotnik in ne kvadrat) smo rešili z matematično 2-dimenzionalno interpolacijo.

Pripravili smo knjižnico računalniških programov za obravnavo (obdelavo in shranjevanje) celične organizacije prostorske baze podatkov. Celično organizacijo prostorske baze podatkov imenujemo organizacija, kjer je osnovna prostorska enota mrežna celica (kvadrat), za katero imamo določene vektorske podatke. Namen programske knjižnice je poenostaviti programiranje aplikacij za vnos podatkov, analize (statistične, prostorske ...), modelov (projekcije, simulacije ...), ter za izpis rezultatov (grafični prikazi, tabele, diagrami ...). Knjižnica vsebuje programe za:

* ustvarjanje, odpiranje, zapiranje, uvoz in izvoz podatkov iz baze podatkov (datoteke C in datoteke CF),
* tiskanje podatkov z vrstičnim tiskalnikom (GRAFC7, GRAFC8),
* izris podatkov z risalnikom Versatec (GRAFCP).

Programe smo večinoma pisali v programskem jeziku FORTRAN (FTN), nekaj pa tudi v COMPASS in PASCAL 6000. Za upravljanje s podatkovnimi bazami smo uporabili za dostop do podatkov metodo naslavljanja do besed (word addressable) v okviru operacijskega sistema NOS/BE 1.3 na računalniku CYBER 72.

*Slika 4: Izpisi zemljevidov občine Celje, vir: Zakrajšek, F. et al: Informacijska zasnova občine Celje ob pripravi prostorskega plana. Ljubljana: Urbanistični inštitut Republike Slovenije, 1979*

5. Primer metod: ekspertni sistemi

V tem obdobju smo razvili vrsto postopkov za ustvarjanje in posodabljanje baze podatkov ter za njeno uporabo in analizo:

* osnovni algoritmi, ki obravnavajo segmentno usmerjene baze podatkov (npr. točka v poligonu, minimalna pot), priprava eksperimentalnega softverskega sistema za upravljanje (posodabljanje, pridobivanje) segmentno usmerjenih baz podatkov,
* modeli projekcij prebivalstva in simulacij (»metoda preživetja kohorte«),
* modeli prostorske interakcije (modeli tipa Lowry, ki temeljijo na principu maksimiranja entropije z omejitvami neenakosti),
* zaposlitveni model za oceno potrebnih delovnih mest na podlagi projekcije prebivalstva,
* stanovanjski model za oceno potrebnih stanovanj na podlagi rasti prebivalstva in obstoječega stanovanjskega sklada,
* projekcije prebivalstva in simulacije za več območij (»metoda preživetja kohorte«), prekrivanje zemljevidov, mrežni zemljevidi različnih fizičnih indikatorjev (npr. vrsta tal, relief, raba zemljišč) so prekriti s kombinacijo razmerij »AND« in »OR«,
* mrežni simulacijski model prispevnega območja za lokacijo lokalnih storitev (npr. osnovna šola, vrtec, zdravstveni dom).

Že zelo zgodaj smo ugotovili, da je uporaba na pravilih temelječih ekspertnih sistemov ali sistema avtomatskega učenja v urbanističnem in prostorskem načrtovanju zelo obetavna. Zbrali smo nekaj primerov uporabe ekspertnih sistemov na sorodnih področjih, prepoznali potencialne priložnosti v prostorskem načrtovanju in pregledali razpoložljivost obstoječe programske opreme za končne uporabnike. Po ponovnem pregledu zbranega gradiva smo razvili več prototipov ekspertnega sistema, imenovanega UEXPERT (slika 5).

*Slika 5: Primer prototipa sistema UEXPERT. Vir:* Zakrajšek, F. *Možnosti uporabe ekspertnih sistemov v lokacijskem postopku*. Ljubljana: Urbanistični inštitut SR Slovenije, 1985

6. Implementacije: občini Celje in Ljubljana

Po mojem mnenju ima sodelovanje pri projektih implementacije dva namena – svetovati in pomagati pri dejanskem razvoju informacijskega sistema za urbanistično in/ali prostorsko načrtovanje na določenem območju ter pridobiti praktične izkušnje za nadaljnje raziskovalno delo in pripravo pravnih aktov na področju prostorskega načrtovanja. V obdobju 1970–1985 smo implementirali naslednji strateški načrt informacijskih sistemov (ISSP):

* Mestna občina Celje (60.000 prebivalcev, 250 km2); vzpostavljeni sta bili dve bazi podatkov. Najprej »osnovna baza podatkov« s hišno številko kot osnovno georeferencirano enoto, s katero so povezani mikropodatki iz več registrov (prebivalstvo, stanovanja, poslovni prostori, vozila). Nato pa še »baza podatkov načrtovanja«, kjer je bila osnovna georeferencirana enota mrežna celica (100 x 100 m) in kjer so podatki zajemali zbirne podatke iz osnovne baze podatkov in neposredno zajete podatke o fizičnih prostorskih značilnostih (nadmorska višina, naklon pobočja, orientacija, stabilnost tal, kmetijska zemljišča, gozdna zemljišča, zavarovana območja itd.). Baza podatkov je bila uporabljena pri pripravi prostorskega načrta Mestne občine Celje.
* Občina Ljubljana Center (32.000 prebivalcev, 5 km2); pri oblikovanju vsebin ljubljanskega informacijskega sistema smo se držali splošnih načel ISSP, upoštevali pa smo tudi naslednje posebne zahteve: določena območja v občini Ljubljana Center (CBD, območja prenove), poleg urbanističnega načrtovanja smo nudili podporo tudi pri določenih administrativnih postopkih (npr. postopek pridobitve lokacijskega dovoljenja), v največji možni meri smo uporabili obstoječe vire podatkov (npr. podatke iz popisa prebivalstva iz leta 1981, registra poslovnih prostorov, registra teritorialnih enot). To je bil pilotni izvedbeni projekt vzpostavitve registra stavb. Informacijski sistem je vključeval zelo podrobno bazo mikropodatkov o poslovnih prostorih, stavbah, gospodinjstvih in prebivalcih (glejte priloženo shemo).
* Mestna občina Ljubljana (200.000 prebivalcev, 500 km2); bistveni del implementacije ISSP Ljubljana je bilo spremljanje dejanske rabe zemljišča v primerjavi z načrtovano rabo zemljišča. Namen tega spremljanja je bil pridobiti trajne in sistematične dokaze o vseh spremembah načrtovane rabe zemljišč, enostaven in javno dostopen vpogled v status ureditve načrtovane rabe zemljišč, trajno in analitično primerjavo med dejansko in načrtovano rabo zemljišč ter enostavno in objektivno vrednotenje posledic predlaganih sprememb načrtovane rabe zemljišča. Osrednji del informacijskega sistema je predstavljala baza podatkov, kjer so bili različni podatki razporejeni po območjih načrtovanja. Informacijski sistem je bil uporabljen pri pripravi prostorskih dokumentov, imenovanih Ljubljana 2000.
* Ljubljanska regija (16 občin, 600.000 prebivalcev, 5000 km2); regionalni lokacijski model je bil v ljubljanski regiji implementiran pri pripravi regionalnega razvojnega plana v okviru projekta Ljubljana 2000. Z uporabo regionalnega lokacijskega modela je bilo pripravljenih in preučenih šest različnih alternativ urbanizaciji: koncentracija, dekoncentracija, sever, jug, stanovanjske soseske in uravnoteženi model.

*Slika 6: Informacijska baza informacijskega sistema za urejanje prostora Mestne občine Ljubljana. Vir: Zakrajšek, F. et al: Informacijska zasnova občine Celje ob pripravi prostorskega plana. Ljubljana: Urbanistični inštitut Republike Slovenije, 1979, Souvan, T., Zakrajšek, F. et al. ISUP: informacijski sistem za urejanje prostora občine Ljubljana Center. Ljubljana: Urbanistični inštitut SR Slovenije: Zavod za izgradnjo Ljubljane, TOZD Urbanizem-LUZ: Uprava za avtomatsko obdelavo podatkov Skupščine mesta Ljubljane, 1985*

7. Implementacija: Gvajana

V letih 1983–1985 sem imel kot del mednarodne svetovalne ekipe v Gvajani priložnost sodelovati pri projektu »Institucionalna podpora sistemu planiranja in projektov v Gvajani«, ki ga je financirala Medameriška razvojna banka. Sam sem vodil podprojekt »Informacijski sistem za planiranje in projekte«. Zadružna republika Gvajana je bila takrat država v razvoju v Južni Ameriki s približno 800.000 prebivalci. Namen razvoja informacijskega sistema za planiranje in projekte je bil zagotoviti podatkovno podporo za nacionalno ekonomsko in socialno načrtovanje ter centralno koordinacijo sektorskega in regionalnega načrtovanja, spremljanje/planiranje projektov in spremljanje podjetij na državni ravni. Informacijski sistem smo razvijali v dveh fazah:

1. Priporočila – diagnosticiranje, koncept informacijskega sistema, informacijski inženiring za informacijski sistem, priporočila za takojšnje izboljšave.
2. Začetna implementacija – razjasnjen koncept informacijskega sistema, pregled obstoječih podatkov, pregled računalniških virov, začetni podatkovni slovar, eksperimentalna baza podatkov – seznam aktualnih projektov, eksperimentalna baza podatkov – podatki popisa prebivalstva iz 1980.

Takrat smo razvili MDBMS2 (Micro Data Base Management System Version 2 – Sistem za upravljanje baz mikropodatkov, različica 2). Šlo je za preprost sistem za upravljanje baz podatkov, napisan v jeziku BASIC za računalniški sistem ID80 (Iskra Delta-80, prej mikroračunalnik ID Partner). Uporabljen je bil za bazo podatkov urbanističnih dokumentov mesta Ljubljana. V Gvajani prepišemo program v jeziku BASIC za Apple IIe. Program je bil uporabljen za upravljanje baze podatkov za seznam projektov v Gvajani (slika 7).

*Slika 7: Shema informacijskega sistema za načrtovanje in projekte za Gvajano. Vir: Zakrajšek, F. Priporočila za razvoj informacijskega sistema za načrtovanje v Republiki Gvajana. Georgetown: Urbanistični inštitut Republike Slovenije, Center za načrtovanje razvoja, 1983*

8. Primerjave in objave

Pri snovanju inovativnih kompleksnih informacijskih sistemov sta potrebna nenehno preverjanje razvoja in implementacija tujih izkušenj na tem področju. Dandanes je to zaradi interneta in dostopnosti informacij veliko lažje. Takrat se je bilo treba opreti na najnovejše strokovne knjige, znanstvene članke, objave s strokovnih konferenc in neposredne obiske posameznih strokovnih ustanov. Da sem dobil informacije, sem moral leta 1983 obiskati nekatere univerze, inštitute in druge evropske ustanove ter narediti oceno. Na koncu tega potovanja sem zapisal nekaj svojih opažanj:

* Lahko trdimo, da so na področju urbanističnih informacijskih sistemov naše raziskave povsem primerljive s tujimi (te so sicer bolj finančno podprte, predvsem pa imajo na voljo boljšo računalniško opremo), zato računamo na izmenjavo določenih rezultatov raziskav.
* Tedanji razvoj je šel predvsem v smeri implementacije manj kompleksnih, cenejših informacijskih sistemov, ki zagotavljajo finančne učinke v obliki prihrankov v kratkem času.
* Glede na stanje informacijskih sistemov, ki dejansko delujejo v praksi na tem področju, so Švedska in nekatera mesta v Veliki Britaniji precej pred nami, medtem ko denimo Škotsko na nekaterih področjih prekašamo z vidika implementacije.
* V državah, ki sem jih obiskal, so dajali velik poudarek uvajanju mikroračunalnikov in razvoju programske opreme za mikroračunalnike. Kljub temu v času mojega obiska še niso imeli pomembnejših izkušenj na tem področju.

Po drugi strani pa je tudi danes še vedno pomembno, da preverimo rezultate raziskav v strokovnih revijah, publikacijah in na konferencah. V tem obdobju smo skoraj vsako leto objavili rezultate raziskav na vsaj dveh konferencah na območju bivše Jugoslavije, na eni s področja prostorskega načrtovanja in drugi s področja računalništva in informatike. Temeljna oblika širjenja informacij je bila tudi objava naših dosežkov; nekaj naslovnih strani si lahko ogledate na sliki 8.

Publikacije smo natisnili v več sto izvodih in jih »prodali« ustanovam po nekdanji Jugoslaviji. Kljub temu, da so bile publikacije napisane v slovenščini, je strokovna javnost zanje pokazala veliko zanimanja.

*Slika 8: Naslovne strani publikacije Informacijski sistem za prostorsko načrtovanje. Letniki: Smernice za razvoj, Koncept sistema, Mikroračunalniška oprema za upravljanje relacijskih podatkovnih baz, Informacijski sistem regionalnih objektov, Možnosti uporabe ekspertnih sistemov v postopku lokacijskega dovoljenja, Metode prostorskega načrtovanja na občinski ravni – projekcije prebivalstva, Možnosti uporabe ekspertnih sistemov pri prostorskem načrtovanju, Izhodišča za izbiro strojne in programske opreme, Sistem kazalnikov načrtovanja za spremljanje rabe zemljišča v Ljubljani, Prenova koncepta informacijskih sistemov.*

9. 50 let pozneje

50 let pozneje smo še vedno zelo prisotni na področju raziskav, razvoja in implementacije računalniške podpore v urbanističnem načrtovanju. Na sliki 9 je na voljo nekaj povezav do našega nedavnega dela.

*Slika 9: 50 let pozneje*

*Slika 9a: 3D Urbanizem, Ljubljana, od leta 2010 dalje Povezava:*

[*https://3durbanizem.ljubljana.si/3DUrbanizem/3DMesto/*](https://3durbanizem.ljubljana.si/3DUrbanizem/3DMesto/)

*Slika 9b: Agent-based Geographical Modeling of Public Library Locations, od leta 2010 dalje. Povezava:*

[*https://www.sciencedirect.com/science/article/pii/S0740818819302890*](https://www.sciencedirect.com/science/article/pii/S0740818819302890)

*Slika 9c: SMUR, Simulacijski model urbanega razvoja, od leta 2010 dalje. Zakrajšek, F. et al. Razvoj in implementacija regionalnega simulacijskega modela za ljubljansko urbano regijo: regionalni simulacijski model. Ljubljana: Urbanistični inštitut Republike Slovenije*

*Slika 9d: Simple Cellular Automata Model for Flood Estimation and Determination of Culture Heritage on Risk, 2018–2022, STRENCH: STRENgthening resilience of Cultural Heritage at risk in a changing environment, Central Europe Programme, 2020–2022*

1. 8. So vrednote v razvoju programja časovno občutljive ali jih lahko shranimo?

mag. Silvana Žorž (digitalna humanistka)

Namen tega prispevka je raziskati, ali je programje mogoče retrospektivno analizirati z vidika vrednot z uporabo metodologije vrednotno občutljivega oblikovanja (angl. *values sensitive design*, VSD), ki v trenutnem etičnem diskurzu na področju računalništva omogoča upoštevanje človeških vrednot pri načrtovanju programja. Prispevek je sestavljen iz tridelnega raziskovalnega procesa, tj. konceptualnega, empiričnega in tehničnega, pri čemer se empirični proces[[36]](#footnote-36) osredotoča na proučevanje človeškega konteksta, v katerem se tehnologija nahaja oziroma se je nahajala, kot je to v našem primeru (Žorž 2017).

* 1. Empirična raziskava VSD – HERMES SoftLab, Slovenija

Družba HERMES SoftLab (HSL) je bila ustanovljena v prvem valu zagonskih podjetij po spremembi takratne družbene ureditve. Oktobra leta 1990 so Rudi Bric, Tomaž Schara, Andrej Kuščer in Zoran Zagorc ustanovili družbo za razvoj programske opreme na podlagi svojih izkušenj, ki so jih med letoma 1984 in 1990 (v obdobju samoupravnega socializma) pridobili pri delu za družbo Hewlett Packard (HP). Glavna dejavnost pa je bila razvoj programja za druge uporabnike. Na vrhuncu leta 2002 je družba štela 700 zaposlenih, med drugim je zaposlovala pomemben odstotek razpoložljivih razvijalcev programske opreme v Sloveniji, poleg tega pa je imela podružnice odprte doma in v tujini.

Rudi Bric je prav zaradi iskrenosti pri razvoju programske opreme sledil določenim vrednotam, ki jih je prenesel tudi v družbo HSL. Ta pošten pristop je temeljil na njegovi izkušnji, ko je med zaposlitvijo v podjetju ISKRA Računalniki leta 1980 (torej precej pred ustanovitvijo družbe HSL) opazoval, kako je tuja družba odpravljala napako v programski opremi.

Zaradi te napake je bila ameriška družba Control Data Corporation (CDC), ki je izdelovala miniračunalnike CYBER-18 in aplikacije programske opreme, položaju moči napram kupcu njihove programske opreme, mehiški družbi DICONSA. Družba DICONSA je morala večkrat plačati za »odpravo napake«, da je lahko še naprej uporabljala aplikacijo. Družba CDC pa napake sploh ni odpravila, in Mehičane raje obtoževala, da so neuki uporabniki, ki napako povzročajo znova in znova. Bric je točno vedel, kje tiči problem in kako ga rešiti. Nazadnje so težavo razrešili inženirji v podjetju ISKRA, vendar je Bric zaradi te izkušnje izgubil zaupanje v odnos nekaterih velikih družb do svojih kupcev. Poleg tega pa se je začel zavedati, da je treba v celoti razumeti vse posledice uporabe razvitega programja, da se ga lahko ustrezno nadzoruje razvojni postopek in zagotovi kakovosten razvoj. Bric je po vseh teh letih ravno zaradi te izkušnje v kulturo družbe HERMES SoftLab vpeljal močno vrednoto odgovornosti pri razvoju, kar je dopolnil z rekom: »Dobro delo ustvarja novo delo.«

Prva največja stranka družbe HSL je bila družba Hewlett Packard (HP), ki je bila v 90. letih sinonim za standarde najvišje kakovosti na področju razvoja programske opreme. Pri HSL so svoj razvojni proces oblikovali na podlagi procesov v družbi HP. Bric je želel ugotoviti, ali so kot podjetje dovolj sposobni, organizirani in strokovno podkovani, da se lahko pri razvoju programja kosajo s komer koli na svetu. Kot je pojasnil Krajnik, je družba HP s inženirskega vidika veljala za zlati standard. Že na samem začetku HSL so bile v proces vključene določene osebe, ki so bile zadolžene za razmišljanje o procesu inženiringa programja in vseh povezanih vidikih, ki bi vodili do razvoja programske opreme, s katero bi lahko samozavestno nastopili na svetovnih trgih. Velik del znanja o podobnih procesih so prenesli iz odnosa med HP in HSL, kar je Bric opisal kot srečno naključje sodelovanja med ustanovitelji družbe HSL, prej zaposlenimi pri družbi HERMES (distributer za HP že v času Jugoslavije), ter višjima raziskovalcema, zaposlenimapri HP-ju prof. Zvonkom Fazarincem in dr. Francem Rodetom (oba sta bila tudi osebna prijatelja ustanoviteljev Billa Hewletta in Davida Packarda). Leta 2000 je HP kupil 5-odstotni delež podjetja. Zaradi poglobljenega odnosa in vizije dobrega dela so se HP-jeve dobre delovne prakse prenesle tudi v delovanje družbe HSL.

Kot je pojasnil Miro Germ, direktor kakovosti v družbi HSL, ki je prišel iz podjetja HERMES Plus, so se prakse[[37]](#footnote-37) družbe HP iz njenega predstavništva v Jugoslaviji prenesle v miselnost ustanoviteljev HSL. HP-jeva priporočila so uporabili pri razvoju poslovne strategije in postopkov nadzora kakovosti prek orodja Total Quality Management (TQM) in knjige Total Quality Essentials avtorja Sarva Singha Soina, ki je bil tudi direktor celostnega zagotavljanja kakovosti pri HP Pacific.

Pristop družbe HP »The HP Way«, ki je zajemal poslovne cilje družbe HP in »kako to tukaj počnemo« (Lacy in Mullins, 2002), je bil prav tako v celoti prenesen, saj je vodstvo delalo za HP Yugoslavia na Dunaju. V tem pristopu je jasno navedeno, da morajo tako družba kot zaposleni naslednjo trditev brezkompromisno upoštevati:

»*Zaposleni na vseh ravneh morajo spoštovati najvišje standarde poslovne etike in razumeti, da je vse drugo nesprejemljivo.*« (HP Alumni, 2022)

To bi lahko razumeli kot razlago sedmega korporativnega cilja, tj. državljanstva, pri katerem je navedeno naslednje:

»*Zaradi zavezanosti k dobremu državljanstvu izpolnjujemo svoje obveznosti s prispevanjem skupnosti in institucijam v naši družbi, ki sestavljajo okolje našega delovanja*.« (HP Alumni, 2022)

Germ se je predstavil tudi Evropski organizaciji za kakovost (EOQ) in s tem pokazal, da je družba HSL v tistem času sodelovala v dialogu o kakovosti, pozneje pa je deloval tudi kot ocenjevalec kakovosti pri različnih združenjih za zagotavljanje kakovosti. Primož Krajnik, direktor podjetja Zaslon (slovenskega proizvajalca bančne programske opreme, ki ga je družba HSL kupila leta 2000), je v tistem času spadal med najmlajše ocenjevalce poslovne kakovosti za Evropski sklad za upravljanje kakovosti. Pri HSL je bil proces zagotavljanja kakovosti zelo podrobno definiran in šele po zaključku tega procesa so lahko začeli s programiranjem. V določenem obdobju je za vsak projekt obstajala projektna nadzorna plošča, kjer so vodje kakovosti spremljali, ali projekt poteka skladno s pravili, določenimi pred njegovim začetkom. Tudi ta proces je izhajal iz družbe HP.

Bric je poudaril tudi dejstvo, da se niso posvečali samo temu, kako vzpostaviti proces, temveč tudi temu, kako zaposlenim približati pravilno upravljanje procesa razvoja programja in jih pri tem tudi voditi. Vse to je prispevalo k oblikovanju inženirske kulture, v kateri so se ekipe zaradi nenehnega učenja, razvoja, izmenjave najboljših praks, popravljanja napak in osredotočanja na izjemno kakovost oblikovale v enakopravne skupine, ki si prizadevajo doseči standardiziran cilj. Hkrati pa so bile ekipe izjemno raznolike, saj so jih znotraj družbe sestavljali člani 14 različnih narodnosti (Tagesspiegel, 1998).

Leta 1998 je družba HSL prejela priznanje Republike Slovenije za poslovno odličnost (PRSPO) in tako postala prva dobitnica takšne nagrade v Sloveniji (Rozoničnik in Valenci, 2017), pa tudi prvo podjetje, ki sledi modelu odličnosti EFQM.

Poleg priznanja s strani industrije in različnih združenj za zagotavljanje kakovosti je bil namen HSL zagotavljati storitve, ki so v najboljšem interesu kupcev – ne v smislu ustvarjanja čim večjega dobička, temveč resnično reševati težave kupcev s programskimi rešitvami in jim zagotavljati neodvisnost, je še dodal Krajnik, ki je bil zadolžen za prenos HSL-jevega načina dela v podjetje Zaslon. Dobiček je bil posledica osredotočenosti na visoke standarde kakovosti na področju razvoja programja.[[38]](#footnote-38)

Kot je pojasnil Zoran Zagorc, eden od ustanoviteljev podjetja HSL, je ta praksa visokih standardov na področju vodenja projektov rezultat znanja, ki so ga razvili ustanovitelji, in praks vodenja nadzora kakovosti, ki jih je vpeljal Germ. To je bila glavna razlika med HSL in drugimi podjetji, kar je pomagalo doseči njihov cilj, tj. zagotavljati visokokakovostno programje mednarodnim strankam, kar je bila tudi začetna vizija ustanoviteljev. S svojo prvo strategijo »dobro delo ustvarja novo delo« so se osredotočili na dokončevanje projektov še pred njihovim rokom in preseganje obljubljenega nivoja kakovosti, kar se je potem odražalo v znanju projektnega vodenja, razvoja in nadzora kakovosti. Družba HSL je imela popoln nadzor nad programjem v smislu razvoja in prednosti pri znanju o razvoju, zato bi lahko programje upravljali, kot bi želeli, to pa bi lahko vodilo do moralno spornih situacij. Miselnost podjetja pa ni bila usmerjena v nasprotovanje strankinim interesom. Kot navaja Zagorc, so bili vsi zaposleni posamezniki pri HSL in ustanovitelji zelo etični, ker je imela takratna družba zelo drugačno etično naravnanost. Omenil pa je tudi trenutke, ko je bila etika zelo pomembna, npr. ko so pri HSL razvijali programje za privatizacijo javnega sektorja. V tem primeru je bila etika v središču razvojnega procesa pri HSL, kar je vodilo do razvoja programja, ki je varovalo vlogo uporabnika in ne lastnika programja. Zagorc se je zavedal, da bi mu lahko tak vpogled pomagal pridobiti nepošteno prednost, kar je zaradi svojega etičnega značaja in poštenosti v celoti potisnil ob stran. To je še eden od znakov moralnih vrednot v podjetju, ki so ga tudi druga podjetja ocenila kot moralnega.

Pomembna je tudi ugotovitev Luke Renka, tehnološkega vodje pri HSL, da je HSL pri razvoju izdelkov večji pomen pripisoval potrebam končnega uporabnika in ne podjetja, ki je programje kupilo. Poudarek je bil na zagotavljanju največje možne vrednosti za uporabnika, zato so se lahko stališča družbe HSL in stranke razlikovala. Družba HSL se je tako prej kot stranka proaktivno prilagodila potrebam trga[[39]](#footnote-39) in uporabnikov, zato je imela s svojimi strankami dolga leta odlične poslovne odnose. Zaradi želje po neodvisnosti strank so bile hibridne ekipe in intelektualna lastnina programja v rokah strank in ne družbe HSL, družba HSL pa je zaradi svojih vrednot in standardov kakovosti lahko razvijala programsko opremo za konkurenčne stranke tako, da so bile ekipe med seboj ločene in niso vedele, na katerih projektih delajo drugi. Med strankami pa niso imeli nobene druge neposredne konkurence HP-ju, za katerega je HSL razvijal programje. Treba je poudariti, da je znotraj podjetja prišlo do trenj med kulturami dela za mednarodne ali lokalne stranke, saj je obstajala razlika med razvojem izdelkov za mednarodne stranke in ponujanjem storitev lokalnim strankam. Pri pojmu odgovorne programske opreme pravzaprav ni šlo za vprašanje ali koncept, saj potrošniška programska oprema takrat še ni bila standard uporabe programske opreme.

Družba HSL je v osnovi pri razvoju poskušala uresničiti vrednote dobrega dela in neodvisnosti, ki bi jih lahko retroaktivno prevedli v vrednote VSD lastništva, univerzalne uporabnosti in avtonomije, kar nam omogoča začetek konceptualne raziskave VSD (Žorž, 2017).

* 1. Trenutni zaključek

Analiza VSD na podlagi študije primera HERMES SoftLab (HSL) je odprla vrsto vprašanj glede analiziranja vrednot pretekle programske opreme.

Prvo je dejstvo, da zgodovinskega ozadja oz. konteksta, ki je oblikoval kulturo prebivalstva v tistem času, ni več. Kot sta omenila Zagorc in Krajnik, so bile vrednote industrije programske opreme v času delovanja HSL vrednote določenega časovnega obdobja in kulturnega ozadja, kjer je vzgoja temeljila na prepričanju, da je dobrobit družbe pomembnejša od dobrobiti posameznika. Po besedah Brica je bil način dela pri HP (*The HP Way*) podoben pristopu do dela v Sloveniji in v Jugoslaviji, v zgodovinskem obdobju, v katerem so ustanovitelji HSL in zaposleni tu tudi odraščali.

Druga težava pa je, da zaradi prodaje podjetja HSL podjetju Comtrade leta 2008 dejansko ne moremo opraviti tehnološke raziskave VSD, saj bi bilo treba dodatno raziskati, katero programje bi lahko izpostavili kot artefakt za predstavitev podjetja HERMES SoftLab.

Iz tega lahko sklepamo, da so vrednote na področju razvoja programske opreme časovno občutljive in jih je težko shraniti zaradi spremenljive narave programske opreme, uporabe, lastništva ter dokumentacijskih praks in standardov.

*Slika 1: Bronasta skulptura v poslovnih prostorih podjetja HERMES SoftLab na Litijski cesti 51 v Ljubljani, delo avtorja Andreja Ajdiča (1997). Na štirih ploskvah angelovih kril je napisano naslednje: Prenos pogodbe družbe Hermes družbi HERMES SoftLab Hewlett Packard, pristop družbe HP »The HP Way«, nagrada za podjetnika leta 1997 in telefonski register zaposlenih iz leta 1996 (vir slike: Rudi Bric).*

* 1. 10. Biografije
  2. Roberto Di Cosmo

Roberto Di Cosmo je računalniški znanstvenik, ki je leta 2010 postal prvi direktor IRILL, pobude za inovacije in raziskave proste programske opreme (Initiative de Recherche et Innovation sur le Logiciel Libre). Po desetletju poučevanja na fakulteti Ecole Normale Supérieure v Parizu je leta 1999 postal redni profesor računalništva na Univerzi Paris Diderot. Aktivno se ukvarja z raziskavami na področju teoretičnega računalništva, zlasti funkcionalnega programiranja, vzporednih in porazdeljenih programov, semantike programskih jezikov, sistemov tipov in linearne logike. Zdaj se osredotoča na nove znanstvene probleme, ki jih je prinesla splošna uveljavitev proste programske opreme, kar je bilo jedro evropskega raziskovalnega projekta Mancoosi. Z velikim zanimanjem spremlja razvoj družbe pod vplivom informacijske tehnologije in je dolgoletni zagovornik proste programske opreme, poleg tega pa je prispeval k njeni uveljavitvi s knjižno uspešnico »Ugrabitev sveta« (Le Hold-Up planétaire, 1998). Leta 2007 je ustanovil tematsko skupino za prosto programsko opremo pri podjetju Systematic, ki je v zadnjih desetih letih spodbudila nastanek več kot 60 odprtokodnih raziskovalnih in razvojnih projektov. Di Cosmo je bil zgodnji član AFUL, združenja francoske skupnosti uporabnikov Linuxa. Njegov najbolj znan prispevek k Linuxu je prva »živa« distribucija Linuxa (od 2000 do 2002), imenovana demolinux, ki je omogočila zagon Linuxa s CD-ROM-a. Leta 2015 je ustanovil pobudo za vzpostavitev univerzalnega arhiva vse javno dostopne izvorne kode Software Heritage, ki jo sedaj tudi vodi.

#### Primož Jakopin

Primož Jakopin se je rodil 30. junija 1949 v Ljubljani v jezikoslovno družino. Oče Franc je bil profesor vzhodnih slovanskih jezikov, ki je objavljal članke na področju slovnice in onomastike, mati Gitica pa je prevedla več kot 50 romanov iz angleščine, nemščine, francoščine, ruščine in poljščine. Ker se je v prostem času najraje ukvarjal z jamarstvom, se je sprva želel vpisati na študij geologije, nato pa je upošteval očetov nasvet in se odločil za študij matematike na Univerzi v Ljubljani, kjer je diplomiral leta 1972. Leta 1981 je na Univerzi v Zagrebu magistriral iz informacijskih znanosti z magistrskim delom [Entropija v slovenskih leposlovnih besedilih](http://www.jakopin.net/primoz/magisterij/index.php), leta 1999 pa je na Univerzi v Ljubljani doktoriral iz teorije informacij z disertacijo [Zgornja meja entropije pri leposlovnih besedilih v slovenskem jeziku](http://www.jakopin.net/primoz/disertacija/index.php). Med letoma 1972 in 1984 je razvil numerični model za ocenjevanje in vizualizacijo kraških jam, ki je bil uporabljen za izračun volumna več jam in udornic v Sloveniji. Rezultati so bili objavljeni na [8. mednarodnem speleološkem kongresu](http://www.jakopin.net/primoz/clanki/1981_MCV/index.php) in [3. evropskem stereološkem simpoziju](http://www.jakopin.net/primoz/clanki/1981_MECS/). Potem ko mu ni uspelo nadaljevati raziskovanja na Inštitutu za raziskovanje Krasa v Postojni, je zamenjal področje svojega delovanja in se posvetil jezikovnim tehnologijam, ki so bližje njegovemu primarnemu področju dela. Poleg tega je bil tudi velik ljubitelj programiranja. S programjem, ki ga je večinoma razvil doma, je uspel ustvariti kariero na področju računalniškega jezikoslovja, in sicer v raziskovalnih vodah kot vodja [Korpusnega laboratorija](http://web.archive.org/web/20190406003335/http://bos.zrc-sazu.si:80/index_en.html) na Inštitutu za slovenski jezik Frana Ramovša vse od ustanovitve leta 2001 do leta 2012 ter kot profesor na Filozofski fakulteti v Ljubljani (1993–2012) in na Fakulteti za Humanistiko v Novi Gorici (2012–2016). Najpomembnejše programje: STAT (statistični paket, Control Data Cyber, 1977), IBIS (programski paket za podatke spremenljive dolžine, ki se uporablja predvsem za obdelavo sekundarnih knjižničnih informacij, kot so podatki o knjigah in člankih, Digital DEC 10, 1981), INES (urejevalnik besedil z bazo podatkov in grafikami, Sinclair ZX Spectrum, 1985, zelo razširjen v Jugoslaviji), STEVE (16-bitni urejevalnik besedil z bazo podatkov, grafikami in namiznim založništvom, ATARI ST, 1987–1992, distribuiran v Jugoslaviji, Nemčiji, državah Beneluksa in na Norveškem, na voljo so bili priročniki v angleščini, nemščini, slovenščini in srbščini/hrvaščini), EVA (urejevalnik besedil z bazo podatkov in jezikovnotehnološkimi funkcionalnostmi, 135.000 vrstic kode v programskem jeziku C, za operacijski sistem DOS, 1992, in operacijski sistem Windows, 1996–), NEVA (iskalnik na osnovi EVA za strežnike Windows, 1999–).

#### Matija Šuklje

Pravnik po izobrazbi in heker po duši, Matija Šuklje se najbolje počuti na stičišču obeh svetov. Od poznih 90. naprej se ukvarja s FOSS (Free & Open Source Software) in pridobil je veliko izkušenj s tega področja v javnem, nevladnem in zasebnem sektorju. Dolga leta je vodil največjo globalno mrežo pravnikov, ki se specializirajo za odprtokodne tematike, in še vedno je njen aktiven član. Trenutno je višji svetovalec za avtorske pravice v mednarodnem podjetju Liferay.

#### Boštjan Špetič

Boštjan Špetič je vodja muzejske zbirke Računalniškega muzeja, ambasador softverske dediščine pri organizaciji Software Heritage, podjetnik in mentor zagonskih podjetij, ki trenutno živi v Sloveniji, vendar je tesno povezan z New Yorkom, kjer je živel večino zadnjega desetletja. Boštjan Špetič je soustanovil, zgradil in prodal dva izdelka v okviru istega zagonskega podjetja Zemanta. Prvi je osebni pomočnik, ki je temeljil na semantičnem spletu in je bil prodan neodvisnemu založniško-tehnološkemu podjetju Sovrn, drugi pa prva programatična oglaševalska platforma (DSP), specializirana za nativno oglaševanje, ki je bila prodana podjetju Outbrain. Špetič trenutno dela v skupini za razvoj podjetja Outbrain kot produktni specialist, je mentor več podjetnikom, član upravnega odbora podjetja Rock Content in vodi Računalniški muzej.

Saša Divjak

Saša Divjak je avtor različnih programskih aplikacij, vodil in sodeloval pa je pri mnogih mednarodnih in lokalnih projektih na področju računalniške avtomatizacije, robotizacije in multimedijskih tehnologij. Dejaven je bil v odborih različnih mednarodnih in lokalnih strokovnih združenj. Bil je vodja Oddelka za elektrotehniko na Institutu Jožef Stefan, pomočnik generalnega direktorja družbe Iskra Delta, dekan Fakultete za računalništvo in informatiko Univerze v Ljubljani ter direktor Laboratorija za računalniško grafiko in multimedije. Predaval je tudi predmete Programiranje 2, Sistemska programska oprema, Operacijski sistemi, Računalniška grafika.

#### Silvana Žorž

Silvana Žorž je magistrica digitalne humanistike (KU Leuven) in magistrica ekonomije (Master in Business and Administration – IBM, Univerza v Ljubljani). V magistrskem delu je raziskovala odkrivanje diskriminatornih faktorjev pri oblikovanju programske opreme, in sicer pod mentorstvom dr. B. Berendta (KU Leuven, Belgija). Magistrsko delo je opravila s *cum laude*, svoje delo pa posvetila vrednostnemu sistemu v oblikovalski metodologiji programske opreme. Sodelovala je s številnimi podjetji (Google, Outfit7 itd.) na področju digitalnega marketinga, razvoja digitalnih produktov, raziskovanja trga in potrošniških navad. Trenutno je kot višja specialistka zaposlena pri digitalni agenciji Red Orbit.

#### Nataša Milić-Frayling

Nataša Milić-Frayling je ustanoviteljica in direktorica podjetja Intact Digital, ki nudi platformo in storitve za gostovanje podedovanih programskih namestitev z namenom omogočanja dolgotrajne berljivosti in uporabe digitalnih podatkov. Podjetje Intact Digital posluje z visoko reguliranimi sektorji, kot so farmacevtska industrija in naravoslovne znanosti, kjer nudi podporo pri doseganju skladnosti s predpisi za regulacijo podatkovne integritete ter rekonstrukciji raziskav in ponovljivosti podatkovnih analiz, vključno s strojnim učenjem in umetno inteligenco.

Nataša Milić-Frayling ima 25 let izkušenj na področju računalniške znanosti in inovacij, vključno s 17 leti izkušenj v podjetju Microsoft Research (MSR). Je avtorica prek 100 raziskovalnih objav in navedena je na desetinah odobrenih patentov. Poleg izvajanja raziskav je pri MSR vodila raziskovalni partnerski program, v katerem je promovirala sodelovanje z inovacijskimi partnerji na področju strateških izzivov, vključno z digitalno prezervacijo in dolgotrajnim dostopom do digitalnih vsebin (projekta EU [PLANETS](http://www.planets-project.eu) in [SCAPE](http://www.scape-project.eu/)). Je zaslužna profesorica na Univerzi v Nottinghamu, kjer je 5 let vodila oddelek za podatkovne znanosti ter prispevala k univerzitetnim raziskovalnim strategijam na področju podatkovnih znanosti in umetne inteligence.

Nataša se v širšem strokovnem krogu aktivno angažira ob kritičnih vprašanjih, ki se porajajo ob razširjeni rabi digitalne tehnologije, in sicer vse od profesionalne etike, zasebnosti in jasnosti zasnove do digitalne zastarelosti in odgovornega inoviranja. Pri UNESCU je članica prezervacijskega pododbora v programu Spomin sveta (angl. *Memory of the World Programme*) in predseduje delovni skupini za raziskave in tehnologijo pri projektu PERSIST. Nataša Milić-Frayling je aktivna članica računalniškega združenja Association for Computing Machinery (ACM), poleg tega pa je bila svetnica v svetu združenja ACM Europe in predsedovala je izvršnemu odboru ACM Women Europe.

Franc J. Zakrajšek

Franc J. Zakrajšek je matematik, višji raziskovalec, računalniški programer in svetovalec, ki od 70. let dalje razvija geografske informacijske sisteme za urbanizem in sorodna področja. Njegovo delo se osredotoča na inovativne pristope, metode, simulacijske modele in uvajanje umetne inteligence v e-načrtovanje. Do leta 2015 je deloval je kot član slovenske strokovne skupine za digitalizacijo in digitalno prezervacijo, ki jo je ustanovila Evropska komisija. Je začetnik Registra slovenske kulturne dediščine. Sodeloval je v več projektih EU, ki so uvajali geoprostorske informacije in 3D-modeliranje v področja kulturne dediščine, muzejev, knjižnic in arhivov kot del evropske digitalne knjižnice Europeana in razvil vseevropski e-kulturni zemljevid eCultureMap. Vodil je številne lokalne in nacionalne projekte in predaval je pri UNDP, UN-HABITAT in TAIEX. Delal je tudi kot svetovalec za razvojno banko InterAmerican Development Bank ter objavil več kot 500 člankov in prispevkov.

1. GNU91. Splošno dovoljenje GNU, različica 2, 1991. Pridobljeno septembra 2015. [↑](#footnote-ref-1)
2. Inria je javna znanstvena in tehnološka ustanova, specializirana za matematiko in računalništvo, ki je bila ustanovljena leta 1967. Deluje pod nadzorom francoskega Ministrstva za visoko šolstvo, raziskave in inovacije ter Ministrstva za gospodarstvo in finance. [↑](#footnote-ref-2)
3. Glej SWHAP na spletnem mestu https://www.softwareheritage.org/swhap. [↑](#footnote-ref-3)
4. [Dokumentacija ob izdaji Flash Player 32 AIR 32 (adobe.com)](https://helpx.adobe.com/flash-player/release-note/fp_32_air_32_release_notes.html)  [↑](#footnote-ref-4)
5. [Zbirka računalniških programov The Internet Archive](https://archive.org/details/software)  [↑](#footnote-ref-5)
6. [Računalniški muzej, Ljubljana, Slovenija](https://www.racunalniski-muzej.si/) [↑](#footnote-ref-6)
7. [Computer History Museum, Mountain View](https://computerhistory.org/); [Centre for Computing History, Cambridge](http://www.computinghistory.org.uk/); [Seznam računalniških muzejev po svetu - Wikipedia](https://en.wikipedia.org/wiki/List_of_computer_museums) [↑](#footnote-ref-7)
8. [Intact Digital Ltd](https://www.intact.digital/) [↑](#footnote-ref-8)
9. Milić-Frayling, Nataša, in Marija Čubrić. "Executable Archives: Software integrity for data readability and validation of archived studies." MEdnarodna konferenca Digital Preservation 2021 (iPres2021), Peking, Kitajska [↑](#footnote-ref-9)
10. [VRML Virtual Reality Modeling Language (w3.org)](https://www.w3.org/MarkUp/VRML/) [↑](#footnote-ref-10)
11. [Cortona3D Viewers za Windows](https://www.cortona3d.com/en/cortona3d-viewers-windows) [↑](#footnote-ref-11)
12. [Michael Takeo Magruder](http://www.takeo.org/) [↑](#footnote-ref-12)
13. [World[s]: Michael Takeo Magruder: 2006](http://www.takeo.org/nspace/ns018/) [↑](#footnote-ref-13)
14. Zbiralna politika Računalniškega muzeja (interni dokument Računalniškega muzeja), Ljubljana 2019 [↑](#footnote-ref-14)
15. Računalniški muzej sodeluje z mednarodno mrežo strokovnjakov, ki omogoča dostop do različnih strokovnih podskupin, pri delu s slovenskimi računalniki pa sodeluje izključno s slovenskimi strokovnjaki. [↑](#footnote-ref-15)
16. Leta 1981, in sicer po letih iskanja smiselne rešitve, s katero bi preprečili podvajanje razvoja in dela več podjetij v državi. [↑](#footnote-ref-16)
17. Interna raziskovalna dokumentacija Računalniškega muzeja, Miha Urh, 2022. [↑](#footnote-ref-17)
18. Interna raziskovalna dokumentacija Računalniškega muzeja, Miha Urh, 2022. [↑](#footnote-ref-18)
19. M. Hlavaty, Zmogljivosti in uporaba opreme za avtomatsko obdelavo podatkov 1988, Zavod RS za statistiko, Ljubljana, 1990. [↑](#footnote-ref-19)
20. Grobelnik. [↑](#footnote-ref-20)
21. http://eprints.fri.uni-lj.si/3945/ [↑](#footnote-ref-21)
22. https://github.com/tstih/idp-dev [↑](#footnote-ref-22)
23. https://www.racunalniski-muzej.si/wp-content/partner/ [↑](#footnote-ref-23)
24. https://en.wikipedia.org/wiki/Snake\_(video\_game\_genre) [↑](#footnote-ref-24)
25. https://zbirka.muzej.si [↑](#footnote-ref-25)
26. Netpreserve.org International Internet Preservation Consortium General Assembly & Web Archiving Conference 2022 [↑](#footnote-ref-26)
27. Čeprav je izraz že dobro uveljavljen, je nekoliko zavajajoč, saj se lastnina nanaša na zelo otipljive in konkurenčne dobrine, medtem ko se intelektualna lastnina nanaša na nekonkurenčne in neoprijemljive dobrine. Če izdelate kopijo knjige, je s tem ne odvzamete njenemu prvotnemu lastniku. V tem smislu gre za sredstvo, s katerim se umetno ustvarja pomanjkanje, do katerega naravno sicer ne bi prišlo. [↑](#footnote-ref-27)
28. Zaradi številnih nedejavnih subjektov (t. i. »patentnih škratov) je nastalo več patentnih združenj in jamstvenih shem. Na tej točki velja omeniti družbo [Open Invention Network](https://openinventionnetwork.com/) in organizacijo [LoT Network](https://lotnet.com/), ki imata številne člane in se medsebojno dopolnjujeta. [↑](#footnote-ref-28)
29. Da vse skupaj malce poenostavimo, bomo za obe vrsti znamk uporabljali izraz »blagovna znamka«. [↑](#footnote-ref-29)
30. Če se ime blagovne znamke tako široko uporablja, da ga povprečni potrošnik ne povezuje več z določeno znamko/določenim izvorom, to imenujemo oslabitev blagovne znamke in ta izgubi učinek. Na primer izrazi »superga«, »edigs« in »selotejp« se v Sloveniji uporabljajo kot splošni izrazi. [↑](#footnote-ref-30)
31. V ZDA so možnosti za zaščito avtorskih pravic baz podatkov zelo omejene. V praksi se na splošno obe rešitvi izkažeta dokaj podobni. [↑](#footnote-ref-31)
32. V nekaterih jurisdikcijah, kot so ZDA in VB, lahko avtor svoje delo nameni javni domeni in se odpove avtorskim pravicam, vendar to v večinskem delu sveta (vključno s celinsko Evropo) ni mogoče zaradi neodtujljivih moralnih pravic. Ena možnost je uporaba licenc, kot sta CC0-1.0 in Unlicense. [↑](#footnote-ref-32)
33. Zakon o avtorski in sorodnih pravicah. [↑](#footnote-ref-33)
34. Npr. CC0-1.0, Unlicense, MIT, BSD-2-Clause, BSD-3-Clause ali Apache-2.0. Če so patenti še vedno zavarovani, je Apache-2.0 dobra izbira (in v tem primeru se izognete licenci CC0-1.0). [↑](#footnote-ref-34)
35. Urbanistični inštitut Republike Slovenije je ena vodilnih ustanov v Sloveniji in nekdanji Jugoslaviji na področju raziskav, implementacije in svetovanja v okviru urbanističnega in regionalnega načrtovanja, ki je bila ustanovljena leta 1955 in je močno povezana s podobnimi ustanovami po Evropi in ZDA. [↑](#footnote-ref-35)
36. Metodologija VSD v tem delu je sestavljena iz intervjujev z učinkom snežene kepe. Prvi intervju je opravljen z Rudijem Bricem, ustanoviteljem družbe HERMES SoftLab (HSL), ki nam pomaga pri retroaktivni analizi morebitnega obstoja in uporabe vrednot v procesu razvoja programja HSL. [↑](#footnote-ref-36)
37. Te prakse so bile v HSL del petkovih jutranjih sestankov, na katerih so zaposlenim predajali informacije o trenutnem delovanju podjetja, predstavljanju dobrih in tudi negativnih praks, pa tudi prihodnjih korakov. Uvedli so tudi politiko odprtih vrat, ki je vsem omogočila zastavljanje vprašanj. [↑](#footnote-ref-37)
38. Kot je poudaril Bric, je bila vloga dobička v Packardovem načinu dela pri HP izražena tako, da je dobiček najboljše posamezno merilo HP-jevega prispevka k družbi. Ta vir moči podjetja HP dandanes v ameriškem sistemu vrednot zveni zelo »socialistično«, takrat pa je bil zelo podoben HSL-jevemu pogledu na svet. [↑](#footnote-ref-38)
39. V tistem času je bil poudarek na dodani vrednosti izdelka na trgu (izdelek, ki rešuje težavo), ne pa na dodani vrednosti izdelka za uporabnika. [↑](#footnote-ref-39)