1975-2015 ... Ohlédnutí za programovacím jazykem Ada

Stručný přehled českých aktivit pracovníků KS Praha v oblasti jazyka Ada

Ing. Jaroslav Vladík

Následujícími několika příspěvky chceme připomenout aktivity, které souvisejí s činnostmi kolem implementace kompilátoru jazyka Ada pro minipočítače řady SMEP, konkrétně pro počítač typu SM 52/12 v rámci tehdejšího podniku Kancelářské stroje Praha v osmdesátých a devadesátých letech minulého století.

Bylo by obtížné popisovat podrobně celou tehdejší anabázi, kterou podstoupili pracovníci Kancelářských strojů a několika dalších spolupracujících institucí v ČR, včetně pracovníků našich vysokých škol, zejména FEL ČVUT v této souvislosti. Alespoň stručně je možno uvést rámcový přehled tehdejších základních aktivit rozprostřených do více než deseti let:

- Analýza základních charakteristik jazyka Ada
- Sledování standardizace jazyka Ada (ANSI standard v 1983, ISO standard v 1987)
- Sledování validace kompilátorů (2 v roce 1982, 100 v roce 1987, 200 v roce 1988)
- Sledování vývoje kompilátorů od USA přes západní Evropu k nám
- Porovnání jazyka Ada s jazyky C, Pascal, COBOL, FORTRAN
- Rozbor obsahu kongresu o jazyku Ada v Mnichově (leden 1988); předsudky (vojenský jazyk, novinka, drahé, neefektivní, neakceptovatelné)
- Pracovní skupina ISO JTC1/SC22/WG9 – Ada: 13 členů včetně Maďarska a Československa
- Kongres Ady v Madridu (červen 1989); Ada vs. C (programování „ve velkém“ a „v malém“)
- Účast na konferenci Ada Europe v Stockholmu (1990)
- Účast na konferenci konference Ada Europe v Dublinu (červen 1990); národní abecedy
- Projekt Ada9X s ukončením v roce 1993 (ocenění českého příspěvku v roce 1994)
- Referáty o jazyku Ada na seminářích SOFSEM, MOP a na konferencích ČSVTS
- Práce na kompilátoru jazyka Ada v jazyku Ada: spolupráce FEL, KS a SzKI Budapest (podmínka, aby přeložil efektivně sám sebe)
- SSAda P (Support Systém for Ada Programmers): první kompilační systém v zemích RVHP pro počítače SM 52/12
- Ada a její význam pro výuku
- Ženy-programátorky v Adě: Alena Kozáčiková, Milada Kubátová, Marie Reitermannová + v Maďarsku Ildikö Szilagyi a Viki Kiss.
- Učebnice programovacího jazyka Ada vydaná v SNTL (Müller, Plášil, Vladík)
- Řada článků o programovacím jazyku Ada v různých odborných časopisech u nás

Ada pohledem roku 2015

J. Ichbiah, Ing. J. Vladík

Asi není povolanější osoby ke shrnutí charakteristik jazyka Ada, než otec návrhu Jean Ichbiah. A je jistě zajímavé s odstupem třiceti let sledovat, jak v roce 1985 odpovídá na otázky vztažené tehdy k minulosti i očekávané budoucnosti a jaká je z tohoto pohledu dnešní realita. Text volně vychází z článku J. Vladík ve Výběru č.6/85.

Uvedený článek vychází z interview s hlavním tvůrcem návrhu jazyka Ada Jeanem Ichbiahem (viz [9]) a navazuje na úvodní informaci o tomto jazyku (viz [1]). Další dostupná česká literatura o Adě je shrnuta v seznamu literatury. Než předložíme čtenáři několik nejzávažnějších otázek a odpovědí týkajících se Ady, shrňme v krátké časové posloupnosti deseti let (1975-1985) vývoj tohoto jazyka:

1975: stanovení požadavků na jednotný programovací jazyk podporující čitelnost, efektivnost a udržovatelnost rozsáhlých projektů

1976: zhodnocení a zamítnutí dosud existujících (26) programovacích jazyků

1977: analýza konkurzního řízení (16 návrhů)

1979: vítězí návrh nazvaný Ada na počest spolupracovnice Ch. Babbage, dcery lorda Byrona, Ady Augusty Lovelace (1815-1852)

1980: vydána oficiální referenční příručka

1983: vydána upravená definice jazyka Ada jako ANSI standard; validován první kompilátor

1984: definice Ady byla zařazena k odsouhlasení jako ISO standard; validován první kompilátor v Evropě

1985: rok předpokládaného „průmyslového“ rozšíření Ady

Na dále uváděné otázky odpovídal J. Ichbiah, který byl tehdy prezidentem vlastní společnosti Alsys, zabývající se problematikou Ady.

Co bylo důvodem vzniku nového jazyka Ada?

Především snaha o zavedení jednotného normalizovaného jazyka. V řadě organizací a v mnoha projektech vytvářejí schopní programátoři „vylepšení“ stávajících programovacích jazyků, aby učinili svou práci na projektu o něco snazší předpokládajíce, že tím zvyšují produktivitu. Neuvědomují si však (nebo si to nepřiznají), že v následujících dvaceti letech se deset generací programátorů bude muset učit tento dialekt, aby mohli udržovat daný projekt. Eventuální nový programátor pak přichází k programu, který vypadá jako napsaný např. v COBOLU a programátor pochopitelné předpokládá, že programovací jazyk byl COBOL. Zabere mu potom více či méně času než zjistí, že tytéž zápisy představují něco úplně jiného.

2015: Zlaté pravidlo, které má, domnívám se, platnost dodnes. Otázkou však je, zda dnes, jako tehdy, jsou tvůrci programových systémů schopni a hlavně ochotni toto akceptovat.

Proto je v případě Ady kladen tak mimořádný důraz na standardní definici. V procesu standardizace do roku 1983 bylo řešitelským týmem zpracováno přes 7 000 připomínek od stovek expertů z patnácti zemí. Takovou to činnost je pochopitelně nemožné provádět pro více jazyků současně.

Jak tomu bylo s požadavkem na jazyk Ada?

Poprvé v historii programovacích jazyků byla ustanovena předem skupina (pod vedením D. Fishera) specifikující přesně požadavky na nový jazyk, a to na základě konzultací expertů z oblasti průmyslu, vojenství, škol apod. nejen v USA, ale z celého světa. Pouze specifikace těchto požadavků trvala zhruba tři roky a byla upřesňována ve zprávách Strawman (1975), Woodenman (1975), Tinman (1976) a Ironman (1977). Tyto požadavky zejména akcentovaly spolehlivost, čitelnost a udržovatelnost programu, vedle víceméně klasických cílů jako portabilita a efektivnost. Kdyby tyto požadavky byly formulovány v šedesátých letech, pak by zřejmě na první místo kladly portabilitu a efektivnost před spolehlivostí a udržovatelností.

*Ve zprávě Ironman se konstatuje: „**Jazyk musí podporovat snadnou údržbu, musí preferovat snadnou čitelnost před snadným zápisem, tj. musí podporovat průzračnost, srozumitelnost a modifikovatelnost programů před snadností kódování.**" Toto je zcela nový přístup, jelikož po řadu let byly vytvářeny programovací jazyky podporující rychlé programování. Trvalo poměrně dlouho dobu, než softwarový průmysl zjistil, že tento cíl je falešný. Vývoj většího programu může trvat třeba dva roky, zatímco udržován bude třeba dvacet let. A čím je program čitelnější, tím snazší je údržba.*

2015: Tento princip by se dal „tesat". Již před třiceti lety jsme se snažili propagovat principy „průmyslového programování" a toto platí zřejmě dodnes.

Proč dochází ke změně kritérií na požadované vlastnosti jazyka?

Zastaralé jazyky jako FORTRAN, COBOL, PL/1 v sobě odrážejí technologii počátku šedesátých let; nesou v sobě pečeť tehdejších relací mezi náklady na hardware a software. Zatímco tehdy byly jeho náklady na hardware dominantní a byla snaha využít počítač 24 hodin denně, dnes jsou relace přesně opačné – hardware je levný a chceme jej použít k maximální produktivitě programátorů.

2015: Toto platí stále více, neboť cena a výkonnost HW je v relacích tehdy nepředstavitelných.

Můžete se vyjádřit k některým kritikům jako Tony Hoare, kteří označují jazyk za příliš rozsáhlý?

Tito tvůrci většinou chápou složitost v matematickém slova smyslu, což není zřejmě vždy bez výhrad aplikovatelné. Můj pohled na jednoduchost či složitost je ovlivněn spíš architekturou. Lidské myšlení má mimořádnou schopnost porozumět strukturám. Za předpokladu, že chápeme hlavní rysy struktury, je schopné utvořit propojení a vytvořit bezprostřední důsledky.

Z toho hlediska považuji Adu za velmi jednoduchou. Toto konstatování samozřejmě nepopírá nutnost konzultovat referenční příručku v jednotlivých konkrétních případech, zejména při zápisu programů.

2015: Vždy jsem považoval a dodnes považuji Adu za jazyk s natolik logickou stavbou, že zvládnutí na úrovni Pascalu je otázkou krátkého kursu. Využití konceptů modulů či generických jednotek, jako nosných prvků onoho průmyslového programování, již odpovídá složitosti problematiky, k jejímuž řešení jsou určeny. Ale i toto je zvládnutelné vcelku bez obtíží.

Na vlastní kůži jsem zažil, že ten, kdo překonal určitou bariéru, si programování v Adě doslova zamiloval.

Očekáváte, že definice Ady je zmrazena na řadu let?

Vydání ANSI standardu v roce l983 lze skutečně označit za okamžik zmrazení. Očekáváme však, že zhruba v pětiletých intervalech budou prováděny revize, jelikož jazyky, které nejsou aktualizovány, jsou mrtvé (latina, sanskrt). Výpočetní technika a její potřeby se vyvíjejí a Ada není výjimkou. Okolo roku l990 lze tedy očekávat revizi jazyka.

2015: Toto je zajímavá prognóza, kdy nová definice Ada93 vznikla opravdu počátkem devadesátých let. A poté ještě následovaly definice v letech 2005 a 2012.

Předpokládáte, že půjde o poměrně malé změny?

Ačkoliv je ještě předčasné něco podobného předvídat, jsem nakloněn předpokladu, že půjde skutečně pouze o malé změny. Vedle změn však bude trvalá potřeba poskytovat přesné či zpřesněné interpretace. I když jsou definice podávány s maximální přesností, dříve nebo později (podobně jako v přirozeném jazyce) se objeví problém, který je třeba rozřešit.

2015: Již v Ada93 byly vzaty v potaz hlavní výhrady k původní definici, zejména v konceptu procesů (Pozn. O dalších definicích a změnách již nejsem informován).

Není to neobvyklé mít standard jazyka tak záhy?

Francouzský humorista Pierre Dac uvedl toto „zlaté pravidlo": „Nejprve střílej, potom zaměř a přemýšlej později". Přesně tímto pravidlem se řídil dosud vývoj programovacích jazyků. Byl navržen jazyk, pro nějž byla vytvořena řada kompilátorů, a poté se zjistilo, že různé kompilátory neimplementují jazyk konsistentně. Teprve pak se formuloval standard. Jedná se tak o typické provádění věcí v obráceném pořadí.

Ada představuje první historický případ, kdy byly věci prováděny ve správném pořadí. Nejprve byl vytvořen návrh, poté byl jazyk standardizován, následovalo vytvoření prostředků pro validaci a na závěr se objevily kompilátory. Uživatel má tak záruku, že validovaný kompilátor interpretuje jeho program konzistentně. Validace totiž sleduje nejen úplnost jazyka (nejedná-li se o podmnožinu), ale – do jisté míry- taky kontroluje, zda se nejedná o nadmnožinu.

2015: Toto je opět nadčasové. Pojem „validace kompilátorů" byl klíčový pro zajištění přenositelnosti programů mezi různými počítači.

Co zejména přispělo k vašemu vítězství ve vypsaném konkurzu?

Nejpodstatnější bylo, že jsem odstartoval pět let před mými soupeři. Navíc u námi navrženého jazyka byl řadou lidí oceňován estetický faktor, který do jisté míry chyběl matematicky pojatému návrhu největšího konkurenta Ben Brozgola.

2015: Ano, Ada je půvabná.

Kterého konceptu jazyka si sám nejvíce ceníte?

Asi to bude koncept modulů, který vytváří jednoznačné oddělení viditelných částic (interface mezi uživateli) a těl modulů (doména implementace). Představme si například hodinky, na něž můžeme pohlížet jako na adovský modul. Má soubor operací, které můžeme používat, aniž bychom znali konstrukci hodinek (např. NASTAV_ČAS, UKAŽ_ČAS...) a musí mít taky mechanizmus (neviditelně uzavřený v pouzdře), který zajišťuje požadované operace.

Obecné pravidlo Ady pak říká, že uživatelský pohled je koncentrován do specifikační části (nemusí se jednat pouze o modul) a implementace je pak dána odděleně (logicky nebo dokonce fyzicky).

Každý může posoudit bezprostřední dopad tohoto principu na každodenní programátorskou praxi. V projektu se jistý tým dohodne na službách poskytovaných různými moduly a na základě této dohody mohou jednotlivé týmy pokračovat v projektu, využívajíce dohodnuté služby, zatímco jiné subtýmy tyto dohodnuté služby implementují, programují těla modulů, (tj. vyrábějí hodinové strojky).

2015: Toto podepisuji dnes a znovu! Na základě těchto principů jsme založili naši spolupráci mezi Kancelářskými stroji, FEL ČVUT a SzKI v Budapešti. Tehdy, kdy jsme programovali kompilátor Ady v Adě, to byla ohromná praktická škola, kdy při společných schůzkách jsme stanovili specifikace (propojení) a „doma“ pak implementovati těla. Vše se pak společně překládalo a ladilo. Bylo to efektivní a až neuvěřitelné.

Jak vypadá použitelnost a používání Ady z hlediska roku l984?

Podobně jako v jiných případech vzbuzuje nová technologie velký zájem a tak tomu bylo v roce l980. Poté se obvykle ke slovu dostávají kritici a skeptici tvrdíce, že technologie je příliš složitá a nebude nikdy normalizována. Myslím, že po validaci prvních kompilátorů bylo v roce l984 toto období již za námi.

První validovaný kompilátor (pomalý interpret, mající spíš historickou hodnotu) pochází z newyorské univerzity. Ještě v polovině r.l983 byl validován kompilátor již průmyslového charakteru, a to společně firmami Rolm a Data Generale (pro šestnáctibitový minipočítač Eclipse). Další validovaný kompilátor byl vytvořen firmou Western Digital pro jejich mikropočítač. Koncem roku l984 mělo existovat okolo dvacítky validovaných kompilátorů. Zejména stojí za zmínku dva vládou podporované projekty v USA u firem Softech a Intermetrics, které vytvářejí kompilátory včetně podpůrného prostředí (např. pro počítače DEC VAX, IMP 370). Naše firma Alsys vyvíjí kompilátory pro minipočítače a existuje celá řada dalších firem pracujících v této oblasti, jelikož se očekává velká poptávka na trhu.

2015: Zde je (poprvé) nutné připustit, že tehdejší optimismus se v tomto směru nenaplnil. A dosud mi není úplně zřejmé, kde byl hlavní problém?

Rozšíří se používání Ady?

Jediná věc by mohla zabránit používání Ady, a to absence kompilátorů. Ada bude vysoce žádanou v průmyslu a orientace na ni bude stále větší. Naše firma se soustřeďuje stoprocentně na soukromý sektor, který považujeme za nadmnožinu sektoru vojenského. Ostatně, kdo by dnes řekl o COBOLU, že to je původně vojenský jazyk?

2015: Tak toto tedy tak úplně nevyšlo. Z ne zcela zjevných důvodů se „průmysl“ (až na výjimky) v Adě nezhlédnul.

Proč byste doporučil Adu vedoucím pracovníkům?

Výhody spočívají ve více aspektech. Vedoucí pracovník se zajímá především o programátorskou produktivitu, spolehlivé programy a snadnou udržovatelnost. Požaduje také, aby mohl nového pracovníka bez velkých průtahů zapojit do údržby hotového díla, aniž by se roky prokousával do jeho podstaty. Chce také (i když snad v míře menší než dříve), aby programy byly efektivní. Všechna výše uvedená přání může Ada pokrýt.

Vedoucí pracovník chce mít „portabilní programátory", aby mohl (dle potřeby) zapojovat jednotlivé pracovníky do projektů. Používá-li se více jazyků, vznikají, samozřejmě, problémy s jejich převodem a zaškolováním.

2015: Toto naopak platí stále, ale nebere se na to příliš ohled.

Redukuje Ada podstatně softwarové náklady při prvotním zápisu programů?

Ada je orientována na snížení nákladů pro celý životní cyklus programového produktu, což nemusí nutně znamenat snížení nákladů na kódování (naopak, může tyto prvotní náklady zvýšit). Co vám je však platné snížení nákladů na pořízení programu, když promarníte mnohem více času (peněz)na ladění, údržbu či aktualizaci.

Při programování v Adě musí programátor více namáhat svou mysl než např. ve FORTRANU, musí důkladněji promyslet problém, který řeší, a tudíž třeba stráví více času při pořizování programu. Avšak jeho program má výrazně větší šanci být správný, zabere méně času při ladění a zejména šetří náklady později při údržbě a aktualizacích.

2015: Opět nadčasové konstatování, náklady na údržbu SW byly a jsou odhadovány na mnohonásobek nákladů na pořízení SW.

Musí být programy v Adě vůbec odlaďovány?

Ano, ale vyžaduje to mnohem méně úsilí než dříve, jelikož více chyb je odhaleno již při kompilaci.

2015: Také toto mohu potvrdit z výše zmíněné spolupráce s maďarskou skupinou.

Jaký mají vztah k Adě lidé z oblasti zpracování dat?

Jak jsem očekával, byl takřka okamžitý zájem o Adu v oblasti „inženýrské". Co mě však překvapilo, byl velmi brzký zájem o Adu také u programátorů v oblasti zpracování dat- tam jsem jej předpokládal až o hodně později.

2015: Tak to se nepotvrdilo prakticky vůbec.

Očekáváte, že Ada bude nahrazena jiným programovacím jazykem v nejbližších deseti či dvaceti letech?

Myslím, že Ada bude eventuálně nahrazena jinými prostředky pro formulování problémů na počítačích. Technologický vývoj opět pozmění ekonomické faktory. Ada je dítětem dnešní doby a stále tedy akcentuje faktor efektivnosti. Za třicet let bude již třeba možné za stejné náklady vyrábět počítače, které budou milionkrát rychlejší, a to, co se nám dnes jeví jako ekonomické, může být v budoucnu neekonomickým. Efektivnost Ady je (v kontextu osmdesátých let) chápána ve smyslu času počítače, který je nezbytný pro provedení jistých operací a takto pojatá efektivnost nemůže, samozřejmě, platit neomezeně.

2015: Vývoj HW je snad ještě rychlejší, než se předpokládalo. Paměť dnes již nehraje žádnou roli.

Má smysl, aby se profesionální programátor učil Adu?

Profesionální programátor by měl určitě vědět o Adě, proniknout do jejich konceptů, a jakmile je to pro něj možné, měl by ji začít používat, je-li přesvědčen, že je vhodná pro řešení jeho problému. Aby mohl fundovaně posoudit její vhodnost, musí sám sobě kriticky přiznat, zda do hloubky (nikoliv nutně do detailu) obsáhl její potenciální možnosti.

Kolik času zabere zvládnutí Ady?

Učení znamená přechod z jednoho stavu znalostí do jiného a nezávisí tudíž jen na koncovém stavu, nýbrž také na stavu počátečním. Je to podobné jako při studiu např. angličtiny. Doba studia závisí (odhlédneme-li od osobních charakteristik) na tom, zda znáte příbuzný jazyk nebo více příbuzných jazyků (francouzštinu, němčinu) a také na tom, zda chcete zvládnout pouze běžnou konverzaci nebo pracovat jako profesionální tlumočník.

Lidé, kteří již zvládli strukturovaný programovací jazyk jako např. Pascal, se zřejmě naučí Adu dříve než např. programátor v jazyku FORTRAN. Pascalský programátor zvládne Adu v úrovni Pascalu zhruba během týdne a je pak otázkou několika dalších týdnů, aby se naučil využívat propracovanější techniky Ady.

2015: Sám jsem vedl několik kursů Ady a mohu potvrdit, že pro programátory nebyl problém tento jazyk zvládnout.

Jste osobně spokojen s Adou jaká je dnes?

Nyní, když jsme Adu dokončili, musím říci, že jsem opravdu spokojen, zejména s celkovou architekturou jazyka. Vidím ji jako katedrálu, v níž se všechny architektonické linie harmonicky doplňují. Kdybych musel dělat návrh znovu, nic bych na něm nezměnil.

2015: „Dnes“, tedy zhruba po třiceti letech se zdá, že z důvodů, které by asi zasloužily samostatnou analýzu, je tento jazyk na pokraji zájmu profesionálních programátorů. (Krátký přehled, viz odstavec “Poznámky editora“ za tímto příspěvkem).

V čem jste se při vývoji Ady poučil?

Naučil jsem se, že tým, je-li motivován, může dosáhnout fantastických výsledků. Naučil jsem se, že je možné vyvinout celky jako je Ada s velkým počtem lidí roztroušených po celém světě, využijete-li pro komunikaci technologii sítí. Přes tyto velké počty lidí však musí podobný projekt být navržen a veden jednou silnou osobností.

2015: Toto opět můžeme jednoznačně podepsat v souvislosti se zmiňovanou spoluprací několika týmů (včetně maďarského) na kompilátoru Ady, který byl v Adě programován.

Doplňující otázku zodpověděl v roce l985 hlavní řešitel projektu Ada v ČSSR Ing. J. Vladík.

Jaká je minulost, současnost a budoucnost Ady v RVHP a v ČSSR?

Ada představuje tak silnou vývojovou tendenci v celosvětovém měřítku, že ignorovat ji, by bylo přinejmenším krátkozraké. Projekty Ady jsou rozpracovány v řadě zemi RVHP (vedle ČSSR v LMR, NDR, PLR a SSSR). A pro konzultační a koordinační činnosti v rámci RVHP byla založena Pracovní skupina Ada (RG20) při Komisi pro problematiku výpočetní techniky (KNVVT) jednotlivých AKADEMIÍ VĚD. Tato skupina se schází zhruba jednou za rok a řeší problematiku jednak v oblasti jazyka Ada, jednak z oblasti tvorby kompilačních systémů pro tento jazyk.

Kompilátory se řeší pro celou škálu počítačů JSEP a SMEP, a jelikož jsou často při řešení používány diametrálně odlišné přístupy (od interpretů přes generátory pascalských či assemblerovských programů až po generátory cílových kódů) bude zřejmě později možnost výběru, ať již podle kritérií efektivnosti nebo úplnosti kompilátorů.

Z hlediska stavu v roce l985 jsou nejdále v řešení v MLR, kde přerušili práce na interpretu pro počítače SMEP a dokončují kompilátor pro počítače typu IMB a Siemens. Jelikož v ČSSR je řešení kompilátor pro počítače SMEP, je projednávána možnost nákupu licence maďarského kompilátoru pro řadu JSEP.

V ČSSR pracuje na projektu Ada poměrně velký pracovní kolektiv, jehož analytickou páteř tvoří zkušení pracovníci FEL ČVUT a větší část programátorských kapacit poskytují KS Praha a KS Vsetín. Významnou měrou přispívají k řešení také pracovníci UJEP Brno.

V roce l985 má být dokončen bázový kompilátor ABC (Ada Beginning Compiler) pro SM 4-20 s těmito omezeními:

1. *nebudou pokryty koncepty procesů, generických jednotek a specifikace reprezentací.*
2. *bude omezeno používání dynamických deklarovaných proměnných, odvozených typů a typů fix.*
3. *Bude dostupná pouze omezená verze správy separátní kompilace.*

V letech 1986-87 by pak měl být tento bázový kompilátor doplněn do úplného tvaru a rozšířen na nové typy počítačů SMES, zejména s větším rozsahem operační paměti. Předpokládáme totiž, že právě minipočítače s větším rozsahem paměti (eventuálně mikropočítače) budou hlavní doménou pro využívání Ady.

Tak jako je očekáváno hromadné rozšíření Ady v západních zemích v roce l985, očekáváme tuto situaci v našich podmínkách okolo roku l988. Devadesátá léta by pak měla být ve znění dominantního používání Ady v celosvětovém měřítku.

Podobně jako J. Ichbiah se domnívám, že jedině absence kompilátoru Ady by mola zabránit jejímu používání a rozšiřování v RVHP. Pro nejbližší (přechodné)období může limitovat její používání omezený rozsah operační paměti dosud dostupných minipočítačů, jelikož Ada, jako nejprogresivnější prvek v rozvoji SW vyžaduje

přiměřeně progresivní HW základnu. Zde se do jisté míry negativně projevuje skutečnost, že náš skluz za celosvětovým vývojem je v tom konkrétním případě v oblasti programovacích prostředků nepoměrně menší než v oblasti prostředků technických.

2015: V ČR byla odborně-informační úroveň po řadu let udržována na prestižních seminářích SOFSEM a MOP (viz řada odborných přednášek na toto téma). Také na mezinárodním poli jsme si udržovali (spolu s Maďary) přiměřenou prestiž, byli jsme účastníky pracovních skupin, zejména pracovní skupiny ISO JTC1/SC22/WG9, která se zabývala aktualizovaným standardem Ada9X. Je pro mne ctí, že náš přínos (zejména v oblasti generických jednotek, který byl předmětem mé aspirantské práce) byl na úrovni této skupiny v roce 1994 oceněn „Uznáním za přínos pro pracovní skupin WG9".

V dalších letech pak však došlo, bohužel, k poklesu zájmu o Adu, a to počínaje mezinárodní komunitou, přes zmíněné semináře až po fakulty a maďarskou skupinu.

Závěr z pohledu roku 2015

Sám se k této problematice dostávám po řadě let, za což jsem vděčný a děkuji kolegyním a kolegům (zejména B. Lackovi), kteří využili příležitosti dvoustého výročí narození Ady Augusty Lovelace a připravili ke konci roku 2015 vzpomínkovou akci. K ní jsem se snažil oživením mých vzpomínek přispět i tímto „resuscitovaným" článkem.