Liet. matem. rink., T. 41, spec. nr., 259–266

Programų, informacinių ir verslo sistemoms kurti naudojamų žinių formalizavimo problemos

Albertas ČAPLINSKAS, Audronė LUPEIKIENĖ (MII)
el. paštas: alcapl@ktl.mii.lt, audrone@ktl.mii.lt

1. Įvadas

Šiuolaikinėms verslo sistemoms aptarnauti plačiai naudojamos informacinės sistemos ir informacinės technologijos. Kita vertus, tiek informacinėms sistemoms, tiek informacinėms technologijoms realizuoti reikalingos atitinkamos programų sistemos. Daugelis inžinerinių procesų, naudojamų kuriant programų, informacines ir verslo sistemas, yra panašaus pobūdžio, jais kuriamus artifaktus galima specifikuoti panašiai, panaudojant to paties pobūdžio žinias. Tai sudaro prielaidas sukurti bendrą infrastruktūrą tokiems procesams realizuoti. Infrastruktūra turi apimti techninę įrangą, informacinį aprūpinimą bei organizacines ir instrumentines priemones. Bendra infrastruktūra įgalintų unifikuoti programų, informacinių ir verslo sistemų kūrimo procesus, padėtų lengviau tarpusavyje susieti verslo sistemas su jose naudojamomis informacinėmis ir programų sistemomis. Tačiau šiuolaikinės instrumentinės sistemos programų, informacinių ir verslo sistemų inžinerijos procesus palaiko gana ribotai ir fragmentiškai, tokioms sistemoms kurti skirtos bendros infrastruktūros kol kas nėra. Dauguma šioje srityje pasiūlytų metodų yra empirinio pobūdžio. Jie sudaryti teoriškai neišnagrinėjus nei kuriamų artifaktų prigimties, nei tuos artifaktus kuriančių procesų.

Vienu iš svarbiausių teorinių uždavinių, sprendžiamų kuriant aptariamo pobūdžio instrumentines sistemas, yra jose naudojamų žinių formalizavimas. Šis klausimas taip pat yra išnagrinėtas nepakankamai. Ypač mažai yra žinoma kaip formalizuoti procedūrines projektavimo žinias, pavyzdžiui, projektavimo strategijas. Tokios žinios yra dinaminės, algoritminio pobūdžio, ir tradiciniai žinių formalizavimo metodai joms formalizuoti netinka. Straipsnyje nagrinėjamos problemos, su kuriomis susiduriama sprendžiant šį uždavinį. Siūloma projektavimo strategijoms aprašyti panaudoti hierarchinių baigtinių būsenų mašinų aparatą. Nagrinėjimai atliekami verslo, informacinių ir programų sistemų kūrimo bendros infrastruktūros nagrinėjimų kontekste [1].

2. Žinių ir informacijos samprata

Kalbant apie duomenis, žinias ir informaciją, paprastai sakoma, kad duomenys – tai formalizuotas faktų aprašas, žinios – duomenų interpretavimo taisyklės, informacija – tai kas

sužinoma, interpretuojant duomenis. Faktais gi vadinami teisingi teiginiai apie realųjį pasaulį. Šitaip, pavyzdžiui, šias sąvokas apibrėžia ISO terminų standartai [2]. Kai kuriems tikslams, pavyzdžiui, nagrinėjant tradicines duomenų bazių teorijos problemas, šitokios duomenų, žinių ir informacijos sampratos visiškai pakanka. Tačiau bendruoju atveju ji nėra nei pakankamai griežta, nei pakankamai išsami, nors ją ir galima formalizuoti, pavyzdžiui, pirmos eilės predikatų logikos ar formų formalizmo priemonėmis.

1 pavyzdys. Panagrinėkime sąrašą, kuriame pateikta informacija apie asmenis ir jų darbuotojus. Tarkime sąrašas pateiktas panaudojant šitokią formą:

Darbuotojai

Eilės nr.	Asmuo			Įmonė		
	Asmens kodas	.....	.....	Įmonės kodas	.....	.....
1	35201010012	...	...	5682089	...	...
...	...	...	...	...	...	...

Nagrinėjant tik užpildytus formos laukus, pirmos eilės predikatų logikos priemonėmis šią formą galima aprašyti reiškiniu:

Asmuo (1, 35201010012)&
Įmonė (1, 5682089) &
Type-of (35201010012, "AsmensKodas")&
Type-of (5682089, "ĮmonėsKodas")&
Type-of (1, "Eilės nr.")&
Instance-of (1, "Darbuotojai"),

kur *Asmuo*, *Įmonė*, *Type-of*, *Inctance-of* yra atitinkami predikatai. Pirmosios dvi reiškinio eilutės aprašo duomenis, paskutinės keturios – tiems duomenims interpretuoti reikalingas žinias. Visas reiškinys yra tam tikras informacijos elementas. Šį pavyzdį būtų galima užrašyti taip pat ir atitinkamu λ-reiškiniu:

$$
\begin{aligned}
&darbuotojai(asmuo,\ įmonė) = \\
&asmuo/įmonė\Big[\lambda xy \bullet \big[darbuotojai(x{:}\ asmens\_kodas, y{:}\ įmonės\_kodas\ )\big] \wedge \\
&is\_a\big[x : asmens\_kodas\big] \wedge is\_a\big[y{:}\ įmonės\_kodas\big]\Big] \bullet, \\
&\lambda xy \bullet [(30000000000 < x) \wedge (y \neq 0)] \bullet
\end{aligned}
$$

Bendruoju atveju žinių samprata yra gerokai platesnė, pavyzdžiui, ji turi apimti ir vadinamąsias „know-how" (procedūrines) žinias. Kaip matome iš pateikto pavyzdžio, tai kaip reikėtų formalizuoti šitokio pobūdžio žinias, lieka neaišku.

Darbe [3] duomenų, žinių ir informacijos sąvokos apibrėžtos bendriau ir griežčiau. Duomenys čia suprantami kaip fundamentaliųjų, t. y., nedalomų, esybių formalizacija. Pavyzdžiui, asmens kodas yra fundamentalioji fizinė esybė. Kita vertus, sąvoka „asmens kodas" taip pat yra fundamentalioji esybė, tačiau ji yra nefizinė esybė, susieta su fizinių esybių (asmens kodų) aibe. Sąvoka „asmuo" taip pat yra fundamentalioji nefizinė esybė. Nuo „asmens kodo" ji skiriasi tuo, kad nagrinėjimų kontekste nėra susieta su jokia fizinių esybių aibe. Kitaip tariant, galima išskirti tris fundamentaliųjų (nedalomų) esybių rūšis: fizines esybes, nefizines esybes susietas su fizinių esybių (jos egzempliorių) aibe, ir nefizines esybes, nagrinėjimų kontekste nesusietas su jokia egzempliorių aibe. Fizinės esybės

formalizuojamos vaizduojant jas atitinkamu *žymeniu*. Nefizinės esybės formalizuojamos vaizduojant jas vadinamosiomis *bendrybėmis*. Jei kalbama apie nefizines esybes, susietas su fizinių esybių aibe, tai ją vaizduojanti bendrybė vadinama *vardu*. Kitos nefizinės esybės vaizduojamos bendrybėmis vadinamomis *daiktais*. Vartojant labiau įprastus terminus, būtų galima pasakyti, kad fizinės esybės vaizduojamos leksiniais objektais, nefizinės – neleksiniais objektais (atributų agregatais). Išskiriami dviejų tipų asociatyviniai ryšiai: išreikštiniai ir neišreikštiniai. Išreikštiniais vadinami ryšiai, kuriuos galima išreikšti skaičiavimus aprašančia taisykle, visi kiti ryšiai yra vadinami neišreikštiniais. Informacija apibrėžiama kaip neišreikštinis asociatyvinis ryšys, siejantis kelis duomenų elementus. Kitaip tariant, gauname trijų rūšių informacijos elementus: žymenų kortežus, domenus ir ryšius. Žinios yra apibrėžiamos kaip išreikštinis asociatyvinis ryšys, siejantis duomenų arba informacijos (arba ir vienus, ir kitus) elementus. Pavyzdžiui, asmens amžius gali būti paskaičiuotas, žinant jo gimimo datą. Šis skaičiavimo būdas ir yra vadinamas žiniomis. Žinios gali būti konkrečios, nusakančios kaip paskaičiuoti konkretaus asmens amžių, arba bendresnio pobūdžio, nusakančios kaip apskritai yra skaičiuojamas asmens amžius. Atsižvelgdami į esybių pobūdį, šitaip gauname trijų rūšių žinių elementus: egzempliorius, laukus ir klasterius.

Remiantis aukščiau išdėstytu požiūriu ir panaudojus λ-reiškinių aparatą galima unifikuoti duomenų, informacijos ir žinių sampratą. Duomenys, informacija ir žinios yra išreiškiami vadinamaisiais unifikuotais elementais. Elementas apibrėžiamas turinčiu vardą triviečiu kortežu:

$$A(S_A, V_A, C_A),$$

čia S_A – elemento semantika,

V_A – elemento reikšmių ribojimai,

C_A – elemento struktūriniai ribojimai.

Išreiškiant elemento semantiką, reikšmių ir struktūrinius ribojimus atitinkamais λ-reiškiniais yra aprašomi duomenų, žinių arba informacijos elementai.

2 pavyzdys. Aprašant žinių elementą jo semantika, reikšmių ir struktūriniai ribojimai yra išreiškiami šitokiais λ-reiškiniais:

$$S_A:$$
$$\lambda y_1^1 \ldots y_{m_1}^1 \ldots y_{m_n}^n \bullet \left[S_{A_1}(y_1^1, \ldots, y_{m_1}^1) \wedge S_{A_2}(y_1^2, \ldots, y_{m_2}^2) \wedge \ldots\ldots \right.$$
$$\left. \ldots\ldots \wedge S_{A_n}(y_1^n, \ldots, y_{m_n}^n) \wedge J(y_1^1 \ldots y_{m_1}^1 \ldots y_{m_n}^n)\right] \bullet,$$
$$V_A:$$
$$\lambda y_1^1 \ldots y_{m_1}^1 \ldots y_{m_n}^n \bullet \left[V_{A_1}(y_1^1, \ldots, y_{m_1}^1) \wedge V_{A_2}(y_1^2, \ldots, y_{m_2}^2) \wedge \ldots\ldots \right.$$
$$\left. \ldots\ldots \wedge V_{A_n}(y_1^n, \ldots, y_{m_n}^n) \wedge K(y_1^1 \ldots y_{m_1}^1 \ldots y_{m_n}^n)\right] \bullet,$$
$$C_A:$$
$$C_{A_1} \wedge C_{A_2} \wedge \ldots \wedge C_{A_n} \wedge (L)_A.$$

Čia $\{A_1, \ldots, A_n\}$ yra sutvarkyta vardų aibė, kur kiekvienas elementas, kurio vardas yra šioje aibėje, vadinamas elemento A komponentu.

J, K – m-argumentų predikatai,

C_{A_i} – komponento, kurio vardas A_i, struktūriniai ribojimai,

L – loginis primityvų, įvestų struktūriniams ribojimams aprašyti, reiškinys.

Darbe [3] išdėstyta teorija įgalina formalizuoti paprastas procedūrines žinias, ji nėra nei pakankamai nuosekli, nei pakankamai išsami. Projektavimo „know-how" žinioms formalizuoti jos nepakanka.

3. Projektavimo žinių formalizavimas

Projektavimo žinios yra nevienalytės. Vienos projektavimo žinios apibūdina projektuojamus objektus, kitos – projektavimo būdus (metodus, strategijas, procedūras ir kt.), trečios – projekto kontekstą (tikslus, ribojimus, naudojamas priemones ir kt.). Panagrinėkime projektavimo žinių formalizavimo problemas išsamiau.

Projektuojant verslo, informacinių bei programų sistemas, projektuotojas privalo turėti šias minimalias žinias:

- apie funkcinę projektuojamos konstrukcijos paskirtį ir norimas tos konstrukcijos nefunkcines charakteristikas (patikimumą, našumą ir kt.),
- apie leistinas projektavimo resursų sąnaudas (laikas, pinigai, kvalifikacija ir kt.),
- apie taikytinas projektavimo strategijas,
- apie kontrolės taškus, t.y. momentus, kuriais tikslinga kontroliuoti tarpinius rezultatus ir, priklausomai nuo kontrolės rezultatų, galbūt kažką perprojektuoti,
- apie projektavimo priemones ir gautų rezultatų aprašymo būdus (projektavimo rezultatų aprašymo kalbą).

Jei projektuotojas naudojasi sistemų atitinkama infrastruktūra, šios žinios turi būti formalizuotos ir saugomos žinių bazėse. Visų pirma turi būti saugomos žinios apie konstrukcijų paskirtį ir leistinas projektavimo strategijas. Žinios apie resursų sąnaudas ir kontrolės taškus gali būti traktuojamos kaip projektavimo strategiją apibūdinančių žinių dalis.

Projektuojamos konstrukcijos funkcines ir nefunkcines charakteristikas galima aprašyti formaliųjų specifikavimo kalbų priemonėmis. Klasikinis specifikavimo kalbų aparatas yra gerai žinomas [4]. Šis aparatas yra pakankamai gremėzdiškas ir sunkiai įsisavinamas. Todėl pastaraisiais metais gimė nauja kryptis – supaprastinti (angl. *lightweight*) formalaus specifikavimo metodai [5], [6]. Jų esmė – formalizuoti tik esminius reikalavimus ir supaprastinti formaliuosius aprašus, struktūrizuojant juos pagal atskirus kuriamos sistemos aspektus. Šioje srityje tyrimai intensyviai tęsiami, išsamiau šių žinių formalizavimo problemų nenagrinėsime.

Projektavimo strategija vadinami nurodymai, kokias projektavimo procedūras ir kokia tvarka privalo atlikti projektuotojas, norėdamas gauti norimą rezultatą (suprojektuoti konstrukciją) [7]. Mūsų nuomone, projektavimo strategijos sampratai formalizuoti, tikslinga panaudoti hierarchinių baigtinių būsenų mašinų formalizmą. Kitaip tariant, į projektavimo strategiją mes galime žiūrėti, kaip į projektuojamos konstrukcijos būsenų (nuo

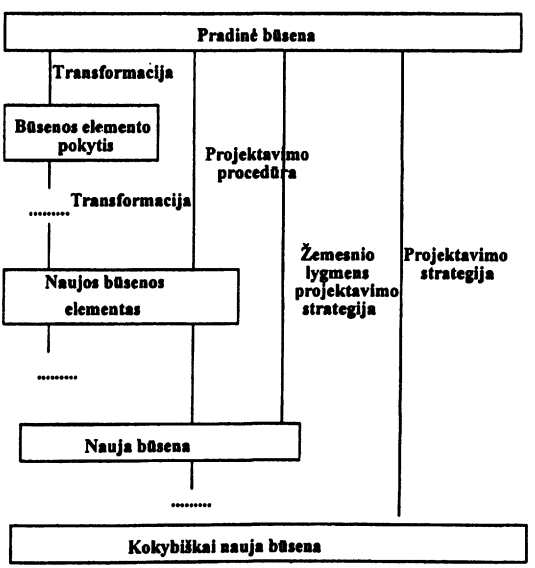
tos konstrukcijos specifikacijos iki jos aprašo projektavimo rezultatų aprašymo kalba) ir tas būsenas keičiančių projektavimo procedūrų visumą. Panagrinėkime konstrukcijos būsenas išsamiau. Visų pirma, galima išskirti kokybiškai naujas projektuojamos sistemos būsenas. Strategija, aprašanti bendriausią sistemos projektavimo būdą, vadinama bendrąja projektavimo strategija. Ji gali būti aprašyta aukščiausiojo lygmens būsenų mašina. Projektuodamas kokybiškai naują konstrukcijos būseną, projektuotojas paprastai taiko dekompozicijos principą, t.y. išskiria tos būsenos struktūrinius elementus ir kiekvieną iš jų projektuoja atskirai. Kita vertus, projektuodamas būsenos elementus, jis stengiasi naudoti transformacinį požiūrį, t.y. transformuoti pradinės būsenos elementą į tikslinės būsenos elementą atlikdamas tam tikras, galimai paprastesnes transformacijas arba, kitaip tariant, įvesdamas tarpines to elemento būsenas (1 pav.). Šitaip gauname hierarchinę baigtinių būsenų mašiną. Transformacijos čia gali būti aprašytos atitinkamais taisyklių rinkiniais, kontrolės taškai – atitinkamomis kontrolės procedūromis. Harelo būsenų mašinų formalizmas [8] ir jo pagrindu sukurtos UML kalbos būsenų diagramos [9] įgalina aprašyti hierarchines būsenų mašinas forma, patogia manipuliuoti tais aprašais kompiuteriniu būdu. Be to, vartojant UML kartu su ribojimų aprašymo kalba OCL, galima formalizuoti ir leistinas projektavimo resursų sąnaudas.

Bendrąją projektavimo strategiją aprašysime trejetu

$$P_0 =< b_{prad}, S, b_{tiksl} >,$$

čia

$b_{prad} \in B$ – pradinė konstrukcijos būsena, kai B – kuriamos konstrukcijos leistinų būsenų aibė,



1 pav. Formali projektavimo strategijos struktūra.

S – konstrukcijos pradinę būseną keičianti bendroji projektavimo strategija,

$b_{tiksl} \in B$ – konstrukcijos tikslo būsena.

Tikslo būsena b_{tiksl} yra kokybiškai nauja (1 pav.), palyginus ją su pradine konstrukcijos būsena b_{prad}.

Bendrąją projektavimo strategiją lemia pasirinktoji konstrukcijos projektavimo metodika. Tai reiškia, kad bendroji projektavimo strategija yra determinuota, t.y., ją sudaranti žemesniojo lygmens projektavimo strategijų aibė turi fiksuotus elementus ir yra sutvarkyta. Pirmojo lygmens projektavimo strategiją apibrėšime šitaip:

$$P_1 =< B_1, I_1, S_1, f_1, b_{prad} >,$$

čia

B_1 – kuriamos konstrukcijos 1-me detalizavimo lygmenyje leistinų būsenų aibė,

I_1 – įvykių, sužadinančių būsenas keičiančias procedūras, aibė,

$S_1 = \{S_{1_1}, S_{1_2}, \ldots, S_{1_s}\}$ – konstrukcijos būsenas keičianti projektavimo strategijų aibė,

f_1: $B_1 \times I_1 \Rightarrow B_1$ – perėjimo iš būsenos į būseną funkcija,

$b_{prad} \in B$ – pradinė 1-mo dekomponavimo lygmens konstrukcijos būsena.

Žemesniojo i-tojo lygmens projektavimo strategija (arba i-tojo lygmens būsenų mašina), kur $i = 2, 3, \ldots, n-1$, o n-bendrosios projektavimo strategijos dekomponavimo lygmenų skaičius, vadinsime šešetą

$$P_i =< B_i, I_i, S_i, f_i, g_i, b_{prad_i} >,$$

čia

B_i – kuriamos konstrukcijos leistinų būsenų aibė,

I_i – įvykių, sužadinančių būsenas keičiančias procedūras, aibė,

$S_i = \{S_i^{k_1}, S_i^{k_2}, \ldots, S_i^{k_m}\}$ – konstrukcijos būsenas keičianti i-tojo lygmens projektavimo strategijų aibė, čia $K = \{k_1, k_2, \ldots, k_m\}$ yra i-tame lygmenyje reikalingų projektavimo strategijų tipų aibė,

f_i: $B_i \times I_i \Rightarrow B_i$ – perėjimo iš būsenos į būseną funkcija,

g_i: $B_i \times I_i \Rightarrow S_i$ – i-tojo lygmens strategijos pasirinkimo funkcija,

$b_{prad_i} \in B_i$ – pradinė konstrukcijos būsena.

Aibės S_i elementai yra tipizuoti, t.y., tam tikro tipo būsenoje gali būti taikomos tik tam tikro tipo tas būsenas keičiančios projektavimo strategijos. Be to, aibės S_i elementai $S_i^{k_j}$ yra pasirenkami iš aibių $\{S_{i_1}^{k_j}, S_{i_2}^{k_j}, \ldots, S_{i_v}^{k_j}\}$, kas reiškia, kad pasirenkama konkreti projektavimo strategija iš kelių galimų to paties tipo projektavimo strategijų. Parenkant strategiją iš kelių alternatyvų, būtina įvertinti galimas jos realizavimo pasekmes. Alternatyvos pasekmės gali būti nagrinėjamos vadovaujantis tam tikru vektoriniu efektyvumo kriterijumi. Alternatyva parenkama vadovaujantis vartotojo preferencijų (t.y. pirmenybės teikimo) funkcija. Yra pasiūlyta kelėtas modelių, leidžiančių nustatyti preferencijų funkciją bendruoju atveju. Pavyzdžiui, [10] darbe aptariamas modelis, vadinamas adityviuoju, leidžiantis įvertinti suminį projektavimo rezultatą, atsižvelgiant į nustatytą kiekvieno iš atributų svorio koeficientą.

Žemiausiojo, n-tojo lygmens projektavimo strategija yra išreiškiama projektavimo procedūrų visuma. Be to, nusakoma tų procedūrų vykdymo tvarka. Projektuodamas kokybiškai naują konstrukcijos būseną, projektuotojas paprastai išskiria tos būsenos struktūrinius elementus ir kiekvieną iš jų projektuoja atskirai. Projektavimo procedūrą apibrėšime šitaip:

$$PP_n =< Q_n, T_n, h_n, q_{prad} >,$$

čia

Q_n – konstrukcijos elemento leistinų būsenų aibė,
T_n – elemento būsenas keičianti transformacijų aibė,
h_n: $Q_n \times T_n \Rightarrow Q_n$ – perėjimo funkcija,
q_{prad} – pradinė konstrukcijos elemento būsena.

Projektavimo procedūros taikymo rezultatas – į tikslinės būsenos elementą transformuotas pradinės būsenos elementas. Skirtingų elementų projektavimo procedūros gali būti vykdomos lygiagrečiai.

Kontrolės taškams aprašyti hierarchinės būsenų mašinos aprašą reikia papildyti kontrolės procedūromis ir jų parinkimo funkcija.

4. Išvados

Kuriant verslo, informacinių ir programų sistemų projektavimo bendrąją infrastruktūrą, esminiu klausimu yra kokio pobūdžio žinios reikalingos šitokiom sistemoms projektuoti ir kaip tas žinias galima formalizuoti būdu, patogiu manipuliuoti jomis kompiuteryje. Šiuo metu naudojami duomenų, informacijos ir žinių formalizavimo būdai skirti duomenų bei žinių bazėms konstruoti ir yra menkai pritaikyti „know how" pobūdžio žinioms formalizuoti. Vienas svarbiausių projektavimo „know how" žinių tipas yra išsamūs nurodymai, kokias projektavimo procedūras ir kokia tvarka privalo atlikti projektuotojas, norėdamas gauti siekiamą rezultatą. Kaip parodyta šiame darbe, projektavimo strategijas galima formalizuoti panaudojus hierarchines baigtinių būsenų mašinas. Šitoks formalizavimo būdas įgalina aprašyti ne tik pačią projektavimo strategiją, bet ir su ja siejamus kontrolės taškus bei jai įgyvendinti reikalingų resursų sąnaudas.

Literatūra

[1] A. Čaplinskas, A. Lupeikienė, *Sistemų inžinerijos intelektualizavimo problemos* (pateikta spaudai).
[2] *ISO/IEC 2382-28. Information technology*, Vocabulary, Part 28: Artificial intelligence – Basic concepts and expert systems (1994).
[3] J. Debenham, *Knowledge Engineering. Unifying Knowledge Base and Database Design*, Springer-Verlag (1998).
[4] A. Harry, *Formal Methods Fact File. VDM and Z*, John Wiley&Sons (1996).
[5] D. Jackson, Structuring *Z* specifications with views, *ACM Trans. on Software Engineering and Methodology*, 4(4) (1995).

[6] E.M. Clarke, J.M. Wing, Formal methods: state of the art and future directions, *ACM Computing Surveys*, **28**(4), 626–643 (1996).
[7] A. Čaplinskas, *Programų sistemų inžinerijos pagrindai*, II dalis, Matematikos ir informatikos institutas, Vilnius (1968).
[8] D. Harel, Statecharts: a visual formalism for complex systems, *Sci. Computer Program.*, **8**, 231–274 (1987).
[9] G. Booch, J. Rumbaugh, I. Jacobson, *The Unified Modeling Language User Guide*, Addison Wesley (1999).
[10] A. Čaplinskas, D. Dzemydienė, *Kai kurie dirbtinio intelekto metodų taikymo aspektai sprendimų priėmimo sistemose* (pateikta spaudai).

Formalisation of the knowledge used in design of software, information and business systems

A. Čaplinskas, A. Lupeikienė

This paper is devoted to the formalisation problems of design knowledge used in business systems, information systems, and software design. It discusses the notions of knowledge and information, introduces the concept of "know-how" knowledge, and proposes to use hierarchical finite state machines as the unified formalism to specify business systems, information systems, and software design strategies.