UML in Practice

Marian Petre
Centre for Research in Computing
The Open University
Milton Keynes, UK
m.petre@open.ac.uk

***Abstract*—UML has been described by some as "the lingua franca of software engineering". Evidence from industry does not necessarily support such endorsements. How exactly is UML being used in industry – if it is? This paper presents a corpus of interviews with 50 professional software engineers in 50 companies and identifies 5 patterns of UML use.**

***Index Terms*—UML, software development, software design, notation, empirical studies.**

I. Introduction: 'Where's the UML'?

The Unified Modeling Language (UML) has been heralded by many as "the lingua franca" (e.g., [23], [24]) or the "de facto standard" (e.g., Sjøberg, interviewed in [26], [6]) of software engineering. And yet there are others who argue that it is not fulfilling this role, because of issues such as size, complexity, semantics, consistency, and model transformation (e.g., [16], [8], [25]). Budgen et al. [6], in their systematic literature review of empirical evidence about UML, conclude that: "There is little to give confidence that the UML has really been evaluated as an artefact in its own right" and "There are few studies of adoption and use in the field" (p. 387). How exactly is UML being used in industry – is it, in practice, the universal notation that it is intended to be? This paper presents a corpus of interviews with professional software engineers about their use (or not) of UML.

Introduced in 1994, UML arose from the unification of three object-oriented design methods: the Booch Method [5], the Object Modelling Technique (OMT) [22], and the Objectory Method [15]. The UML standard was set and is managed by the Object Management Group (OMG). UML offers a framework to integrate many kinds of diagrams, but it also inherits many interpretations. If we treat 'UML use' as a variable, it would have to be continuous, not discrete. Some people use class diagrams; some scribble sequences on a whiteboard; some use UML for model-driven development. To someone arguing that UML is the 'lingua franca', any such use might be sufficient; but the different uses are not equivalent – they have different purposes and different consequences.

The issues associated with interpreting what it means to 'use UML' are familiar. One informant related a story about attending a workshop for software professionals in which the speaker was a UML exponent from IBM who asked how many people in the audience used UML. Of the 50 or so people in the audience, about 47 raised their hands. The IBM speaker understood this to mean that 47 people had adopted full use of UML "with rigor" (as he later expressed to the informant). In contrast, the informant concluded that probably 45 of the 47 were like him: "selective borrowers" … "who use some of the principles sometimes". The IBM speaker and the informant had very different models of what 'using UML' means in practice, with different implications.

Budgen et al. [6] point out that UML development has been guided more by expert opinion than by empirical evidence or cognitive theory. They call for "more and deeper studies of [UML's] longer term use in the field" (p. 387). The work reported here is based on the notion that understanding the nature of actual UML use is important to the discipline, and that understanding how software professionals 'use' UML can inform the development of software design notations and tools.

The study reported in this paper has its origins in a discrepancy of experience. After conducting empirical studies of software design in industry for decades, the author found recently that some of her papers on design representation were challenged by academic referees who asked: "Where's the UML?" Discussions at conferences such as ICSE and ESEC/FSE reinforced the discrepancy, with delegates surprised or even distrustful that the reported professional software design practice did not include use of UML. The response was predictable: to seek new evidence.

II. Background: UML Use in Industry

For their systematic review, Budgen et al. [6] identified 49 papers published up to the end of 2008 that report empirical studies of UML. The majority of papers reported studies concerning UML metrics (12 papers), comprehension (14.5), and model quality (7.5) (with half values indicating papers addressing more than one focus); only 2 papers addressed adoption per se. They note a preponderance of laboratory experiments – and correspondingly little use of field studies. They identify deficiencies in the evidence base, noting that the reported experiments tend to have a single focus and make extensive use of student participants, and that the few reported studies in realistic settings used relatively simple forms of data collection. They concluded that: "There is therefore a real need for more and deeper studies of its longer term use in the field." (p. 387) There are some case study accounts of experiences employing UML on substantial projects (e.g., [1], [2]), and there are a few surveys of UML use in industry, described below.

Similarly, Grossman et al. [12], in introducing their own web-based survey of UML adoption and use in the software

development community, argued that "Much of the existing literature elating to UML usage focuses on … shortcomings" and that "there is still very little empirical evidence available describing the actual usage patterns or performance impacts of UML." (p. 384) They surveyed only participants who were "directly involved with UML" (p. 385), accessing their 131 informants via various online newsgroups. They report "a wide diversity of opinion regarding UML"; for example, on one hand respondents did not find UML's complexity an impediment to its usage; on the other "Nearly, 62% of those surveyed agreed with the statement that UML is insufficiently specified which allows for misinterpretation." (p. 387) They conclude: "Developers clearly seem eager to use this highly hyped technology…Yet, they are lacking an adequate enough understanding of the technology to determine if it is making any real difference in the way they are performing their development tasks." (p. 396)

Dobing and Parsons [7] report on a survey of 171 UML users (plus 11 who use UML components within another OO methodology) undertaken in 2003 and 2004. They reported that their respondents used a subset of the modelling notations available in UML 1.5: "Only Class Diagrams are being used regularly by over half the respondents, with Sequence and Use Case Diagrams used by about half." (p. 112). Collaboration / communication diagrams were the least popular. They also report disappointing effects of UML on communication, with 55% saying that UML was at best moderately successful in facilitating communication with clients. Explanations given by their respondents for not using UML included that it is: "Not well understood by analysts" and "insufficient value to justify the cost."

Nugroho and Chaudron [19] also argue that "Despite the fact that UML is widely used in practice, little is known about how UML is actually used." Their 2007 survey of 80 professional software developers who use UML (80% from the Netherlands; 56% from 3 companies) focused on "the impact of UML modeling styles on quality and productivity issues from the point of view of software engineers." (p. 91) They found that most UML models are incomplete – they specify only part of the required system elements. Developers tend to focus modelling efforts on the more complex or critical parts of systems. 29% of their respondents reported that problems of understandability of UML models led to problems in implementation 'often' or 'very often'. A further 64% reported that it happened 'sometimes'. Their previous survey [18] focused on synchronization between model and code and found that systematic approaches to maintaining correspondence are rarely used in practice.

Lange et al. [17] conducted a web-based survey of 80 software architects, supported by analysis of UML models in 14 industry case studies. Their study focused on UML use and model quality in actual projects, rather than on its adequacy as a notation. On the basis of the responses, they identified four main classes of problems encountered: scattered information (e.g., design choices dependencies); incompleteness, disproportion (more detail for some parts than others), inconsistency. Additional (contributory) issues included: diagram quality, informal use, lack of modeling conventions.

Forward and Lethbridge [9] surveyed 113 software practitioners in 2007 "to uncover their attitudes and experiences regarding software modelling" (p. 27). UML was identified as the dominant notation in their survey, with 52% usage. They found that modeling tools are used primarily for early design work; code generation is not widely used. "The participants seem to really want to incorporate modeling into their processs, but…at present they are not doing so." (p. 30) The biggest perceived problem of model-centric approaches is keeping the model up-to-date with the code (68% agreement). Their sub-sample analysis is particularly interesting, finding that "…'programmers' are more likely to agree that modelling tools are too 'heavyweight'"; "Participants working on real-time systems are more likely to agree that their organizational culture does not like modelling", and "Participants that generate code from models are less likely to agree that modelling tools hide too many details." (p. 31)

It must be noted that many of these studies focused on software developers who use UML; there is little consideration in the academic literature about how representative that sub-population is of the software development community as a whole. In contrast, two industry surveys by MediaDev and BZ Research addressed the issues of penetration and obstacles.

MediaDev [21] carried out a cross-European survey of 500 developers to investigate the penetration of UML into the marketplace and the usage of UML tools within software development. "The majority said that UML tools are not considered as an important part of their development process. More specifically, 41% of the developers that participated in the survey claimed that they did not regard UML as important to the way they work. 29% regarded the UML tool as important, but emphasized that it was not essential to the development processes. 30% viewed UML as an essential development tool, and that the tool was an important part of their development processes." The survey found that there are conflicting views about the importance of UML even within the same department, and that views and preferences are "highly individual". Cost was an important issue that impeded use.

A survey of 226 developers conducted in 2002 by BZ Research [27] found that "In fact, only about one-third of developers recently surveyed said they use UML – and not a single respondent believes that code generated from models is production-ready." "Why don't those 62 percent of developers use UML? The largest number, 30 percent, said that they didn't see any benefit, while 25 percent said that their tools do not support UML. An almost equal number, 24 percent, said that UML-based modeling was too expensive to implement, while 17 percent said that it's too complex to use. Interestingly, 15 percent complained that the code generated by modeling tools isn't production-ready, while 13 percent said that UML is too complex to learn. Also, 23 percent of respondents said they had additional reasons for not adopting UML, and 15 percent said they didn't know why it wasn't used."

The discourse on UML has been informed largely by expert reflection and opinion rather than empirical evidence. There have been repeated calls for deeper investigation of actual UML use in realistic settings. Studies of use in industry so far (typically reflections on individual cases or surveys) identify both examples of effective use and a number of concerns; the software development community appears divided in its assessment of UML. Whether or not UML is the 'de facto' standard, there remain questions about the extent to which it is used and the nature of its use. The study reported here attempts to add to the body of empirical evidence about actual use in professional practice, addressing an omission identified by Budgen et al. [6] and others, with richer practitioner accounts drawn from interviews.

III. Method

The empirical study reported here is a series of interviews conducted over 2 years with more than 50 practicing professional software developers. Informants were identified with an eye to gathering a broad range of perspectives, from corporate large-scale commercial software developers to independent consultants, and across a variety of application areas. Informants came primarily from countries in Europe and North America, but there were also informants from Brazil, India, and Japan, and many had worked in more than one country. Informants were identified opportunistically, via networks of collaborators, colleagues and contacts – people who could act as 'brokers' for introductions of various kinds: at meetings and conferences, via mailing lists, via social networks such as the Requirements Engineering Specialist Group (RESG) on LinkedIn, and via personal emails. All informants were practising professional software developers in roles ranging from requirements engineering, to software architecture, software development, and quality assurance (and most identified themselves as fulfilling more than one role). Only one informant per company was included in the reported data, reducing the sample size to 50.

The sample is arguably broadly representative of professional software engineering, covering a variety of organizational contexts and sizes, practitioner roles, and application areas. The industries represented included: financial services (including insurance and banking), search engines and browsers, social networking, digital audio, digital video, computer games, automotive systems, aerospace, CAD systems, real-time systems, enterprise software, control systems, telecommunications, web development, software tools, civil service, heritage, medical information systems, retail systems, automation. Company sizes ranged from independent consultancies to global, 'household name' corporations. If anything, the sample may be biased slightly toward informants who had something to say about UML, given that some participation was solicited via mailing lists and social networks. For example, one volunteer replied: "We could probably find a couple of people who use UML for you to talk to." As such, any counts in the paper are offered for the purpose of description and are not held to be statistically characteristic of the whole population of software engineers.

Simple, semi-structured interviews were conducted over the phone, on Skype, or in face-to-face meetings, as convenient. The protocol was straightforward, starting with background questions about the professional's experience, role, organizational context, and software projects. The key question was: 'Do you use UML?' Depending on the response, the second question was either: 'Can you tell me about how you use it?' or 'Why not?' Subsequent questions followed up responses and elicited examples of use of UML or other design representations. When appropriate, the informant was asked if his or her usage was typical of the organization. Hand-written or typed notes were captured for all interviews, and, subject to the informant's preference, some interviews were audio-recorded. Some informants provided actual examples of design representations, within confidentiality agreements.

Discussions at times extended beyond the informants' current practice to past projects, past organizations, or other experiences. At times the discussion distinguished between the use by the informant and the use preferred or mandated by the organization. All accounts of UML use offered by the informants were collected, but a distinction was made in the data collection between the informants' own current use (identified in this paper as 'declared current use') and accounts of their own practice in the past or in other organizational contexts, accounts of organizational preferred practices, or accounts of their colleagues' practices which they have observed directly (identified as 'secondary reports'). This paper focuses on responses to do with current practice but includes, where relevant, discussion on 'secondary reports'.

The analysis was inductive, allowing categories of use to emerge from the data. The initial sorting into 'use' and 'non-use' was obvious. Additional categories were identified in terms of what the informants presented as characteristic of their use. The categories, along with a representative selection of anonymized data, were presented to two experienced professional software developers for independent review, as a form of validation.

IV. Overall Results

Five patterns of use were identified. The numbers in parentheses (repeated in Table I) indicate informants whose declared current usage fits within that category.

1. **No UML** (35/50);
2. **Retrofit** (1/50): don't really use UML, but retrofit UML in order to satisfy management or comply with customer requirements;
3. **Automated code generation** (3/50): UML is not used in design, but is used to capture the design when it stabilizes, in order to generate code automatically (typically in the context of product lines);
4. **Selective** (11/50): UML is used in design in a personal, selective, and informal way, for as long as it is considered useful, after which it is discarded;
5. **Wholehearted** (0/50 – but described in secondary reports): organizational, top-down introduction of UML,

with investment in champions, tools and culture change, so that UML use is deeply embedded.

TABLE I. DECLARED CURRENT UML USE AMONG 50 PROFESSIONAL SOFTWARE DEVELOPERS FROM 50 COMPANIES.

Category of UML Use	Instances of Declared Current Use
no UML	35
retrofit	1
automated code generation	3
selective	11
wholehearted	0

Each will be described and discussed in turn, using representative excerpts from the interviews. The counts address declared current usage only, although information from 'secondary reports' are included in the discussion, where relevant.

V. NO UML

The majority of informants (35/50) do not use UML. One stated categorically that his global corporation "doesn't use UML". There is a consistent pattern to the responses concerning why practitioners do not use UML: they have considered it, often having adopted it for a project or two, and found that it did not offer them advantages over their current, evolved practices and representations. As one informant phrased it, and many others expressed: "What was good about UML was not new, and what was new about UML was not good." Another identified where elements of UML notation originated pre-UML, arguing that use of those prior notations alone does not constitute UML use. Some felt that UML came with too much 'philosophy' or 'ideology', that it required them to make changes to their culture or ethos that were not warranted by adding benefits: "Why would I [use UML]? Doesn't add anything except religion." Although phrased strongly, these opinions were not baseless. Most informants spend time regularly engaging with the literature and investigating proposed methods, tools and representations. A number of informants in this group remarked that they had studied and used UML within formal education. One informant, who summarized that "UML does things we already had ways of doing. The notation doesn't really matter", also set out a specific series of equivalences in his practice for elements of UML (e.g., "flowcharting, activity diagrams (I see them as equivalent) – swim lanes have been around since the 1920s", "ER diagrams (or class diagrams, roughly equivalent)"; "state diagrams – we've been expressing state machines for decades"; "the whole front of identifying context – a context diagram is rather better than trying to do anything with classes").

Some of the informants who do not currently use UML offered specific criticisms. The most frequent were:

A. Lack of Context

Informants remarked that UML deals primarily with software architecture rather than the 'whole' system, and hence that it lacks context. For example: "You don't get context with UML. It assumes: 'If they all do their own bit to the spec it will work.' That's a bad assumption." Another informant reported that his company had worked on variations and extensions of UML to address their needs, before abandoning it for another in-house formalism that addresses the 'whole' system "i.e. software, hardware and how they fit together, along with context, requirements (functional and non-functional), decisions, risks, etc."

B. The Overheads of Understanding the Notation.

Issues of comprehension included both software developers and stakeholders. UML is considered "unnecessarily complex". Several informants reported variations in understanding and interpretation among developers that led to problems. For example, "There are challenges in the formal semantics of UML: e.g., 'align' means different things to different people." In another example: "We worked with a group of people from [another large company], and what they did was rubbish. Their use of UML was so distorted that it was unrecognizable."

Others noted that the complexities of the notation limited its utility – or demanded targeted use – in discussions with stakeholders (including highly technical stakeholders). Two informants gave detailed accounts of past experiences of adopting UML on projects in response to client requests. However, those experiences were described as "disastrous": the clients "…refused to sign it off, just too much detail, and they couldn't make sense of it". This was echoed by a third informant: "The best reason not to use UML is that it is not 'readable' for all stakeholders. How much is UML worth if a business user (the customer) can not understand the result of your modelling effort? He would be asked to sign off something that he isn't able to comprehend…"

A related issue is the intrusion of the notation into the discussion or reasoning process. One informant explained: "UML seems to be very much based on programming concepts rather than analysis concepts", making it problematic for discussions with stakeholders. "If you're talking to a client, you map out the concepts that are there in what their business does; you can use an entity model or an object model. The entity model allows you to more or less map the concepts that someone has directly onto a piece of paper, whereas a class model… some of the concepts they have don't appear in the diagram, and some of the things you end up putting in the diagram are not things the person has told you about. You start making decisions about how you're going to implement classes, inheritance and so on, when you need to be mapping out what their business does." A number of informants remarked on the intrusion of the notation into design discussions with both clients and colleagues, because it required explanation or alignment of interpretation, because the notational choices/constraints were not a good fit for the context, or because the use of UML diverted attention from the primary focus onto notational concerns.

C. Issues of Synchronisation/Consistency.

A number of informants identified issues of synchronization or consistency as a barrier for wholesale adoption of UML. One informant described the issues this

way: "…as mentioned by both Clifford and Geoffrey, UML is a graphical representation. There is no check on consistency, redundancy, completeness or quality of the model what so ever. Modeling a small project may not be a problem, but handling large projects in UML forces you to go over the entire model every time you want to change anything, to see what the consequences for the rest of the model will be." Another expressed it in terms of commitment: "Need to use it all the way, in order to maintain sync."

Several informants volunteered preferred alternatives. For example: "Object models… they're not a compelling way of describing things. I prefer entity diagrams, which are a bit better." Another argued: "There are a number of advantages of CoRE that are not available using UML. The key difficulties are the inability to assess cross-system performance prior to the detail design stage and the ability of domain experts to access information from UML models. Failure to assess system performance early in the design process during the system architecture definition phase leads to increased rework costs." A third reported: "I've previously used SSADM, entity modeling, data flow diagramming, and I'm afraid I've reverted to using those rather than UML. A bit more rigorous, more suited to taking something through from requirements to design."

VI. Retrofit

One informant (1/50), plus three secondary reports, report using only UML because it is demanded by the bill-payer, either management or a client. 'Retrofit' UML use means, by and large, documenting things after-the-fact. These informants report using their own practices during design, and then retro-fitting UML when the design has stabilized, either to satisfy a corporate edict (in a large company where the decision to use UML and the business decisions are separated from design practice) or to satisfy a client who is complying with some form of industry or corporate standard. For example, one informant responded to "Do you use UML?" with "Well, not unless the client demands it for some reason" before clarifying that his relevant examples were in the past, rather than in current practice. In his examples, clients asked for UML for requirements documents or design proposals. Another explained that "Use Case diagrams are primarily used only to please the IT Auditors / compliance documentation." One considered that UML is requested because it "gives an illusion of accuracy". In the context of retro-fitting, the translation into UML is usually handed to a junior member of the team (i.e., it is treated as a mechanical, low-priority task).

VII. Automated Code Generation

In this case (3/50), UML is not used in software design (i.e., not typically during the creative stages of design), but is used to capture the design when it stabilizes, in order to generate code automatically. The three informants who described this usage were all working in product lines or embedded software, all contexts in which the software was a 'sub-system' rather than the main product in its own right, and in which software per se was not the sole business or primary focus of the company. In product line development, UML is used to capture a complete architecture with all of the options and variants in it, so that a selection can be made. This also implies a mapping onto the code base, and using the UML spec to drive the derivation process. One informant reported that his company operates on the assumption that "there will be later releases", and so turn-around on releases was valued above producing optimal or complete software.

VIII. Selective Use

Most (11/50) of those informants who do use UML – and who use it in software design in particular (i.e., in the creative phase, rather than just for documentation or code generation) – use it in a personal, selective and informal way, characterized by one informant as "soft use". Some use it infrequently; for example, one informant described his use as: "Very rarely and very selectively – emphasis on rarely", which he quantified as three to four times in seven years. Others use it regularly. They use it for as long as it is considered useful, after which it is set aside or even discarded. For many, this means that UML features only in early design, when the problem is explored, requirements are elicited, and design alternatives are considered. Sometimes it is for personal use only, as a 'thought tool'. Sometimes it is used to "prototype ideas". More often it is used in design discussions, whether in the "instruction of ideas" (i.e., conveying ideas to others) or in collaborative dialogues. Some find it useful for requirements elicitation with key stakeholders (where the stakeholders tend to be highly technical), then discard the UML diagrams when they are through with the discussions. For other informants, selective UML use carries on throughout the design and development process, with UML representations being included in the technical documentation. Different aspects of selective use are discussed in the sections that follow.

A. UML as a 'Thought Tool'

Many of the 'selective' informants use UML as a 'thought tool': "to help me think about code" … "it's a scratch pad". One informant clarified: "I use the concepts." Another explained: "Architects use UML for context for describing detailed solutions in the relationship between different components and the description of components and how they interact. The formality of the notation helps them think about all the interactions … 'a thought framework' … kept in their documentation, but not carried forward into design."

B. Communicating with Stakeholders

Most of the informants talked about using UML in communication with stakeholders, especially technical stakeholders. As one summarized: "UML – mainly sequence diagrams – is useful for requirements elicitation with key stakeholders." The dominant elements of UML in this context were sequence diagrams and activity diagrams, used to elicit requirements and to consider key behaviors. Several informants pointed out that their UML use varied depending on the context, including the audience, for example: "I change the diagrams for different dialogues: the boss, colleagues, stakeholders". Conversely, another informant articulated the

danger of not varying UML use to adapt to the context: "A diagram for a purpose, re-applied, ends up with a mess."

All of the informants who talked about using UML in communication with stakeholder emphasized the importance of keeping diagrams focused and as simple as possible, e.g.: "It can be very hard to communicate with people, except in a very simplified way." Another informant emphasized that, to be effective for dialogues with clients, UML: "Needs to be used to communicate, not just to represent or as a product in its own right." Another emphasized abstraction and context: "Key diagrams, with structure at a high enough level. Gives them a way in…" Another echoed this approach, while also acknowledging the need to avoid premature implementation decisions: "Full-blown modeling is too much for most cases, and for most people...The aim is getting people to ask tough questions: why; who wants that; are there conflicts that need to be resolved? Using goal modeling all the way through is jumping the gun a bit. You end up pre-judging design decisions and you may easily get them wrong."

C. Collaborative Dialogues

Many informants reported using UML in collaborative dialogues. For example: "Architecture colleagues use it as part of the design description, to make "an implementation proposal." Another reported: "Especially when working with teams: use cases, decomposition help us talk through the project". Another described a large integration project that combined software from two different teams, when UML was useful for presenting "… our own system in class diagrams and sequence diagrams, so people understood the lifecycles of the data." One informant found UML particularly helpful in international collaborations: "UML helps a lot when talking to [international] speakers – it provides a language bridge" allowing participants to "…resolve the ambiguity using lower language skills". In each of these cases, UML is used to provide a common representation from which to drive discussion and build a shared model of the problem context and design proposals – potentially overcoming discrepancies in perspective, history, culture, or language proficiency.

D. Adaptation

Selective users are explicit about taking license with the notation or using it "not by the book". Many echoed the report of one informant, that: "I adapt UML to the task." Many identified variations or annotations that they use. For example: "I use variants of activity diagrams for communication purposes or … as part of the thinking process to map out what needs to be done. I use something that's in-between a use case view and an activity view: the user roles, the things going on in the computer, the things in other places, like the back office of the customer." In another example: "I do a swim laney thing for web applications or showing what's in the front end of the application and the back end." One reason for adaptation was to make explicit the relationships between views: "How can I connect all those pictures … interaction between the pictures?" Another reason is to address perceived deficiencies in UML: "Difficult to represent tasks, threads, processes…"

E. Keeping It Small – Selective Traction

Those who reported using UML most enthusiastically use it in a focused way that narrows the scope and hence keeps the UML artefacts manageable in size and fit for purpose. For example: "UML is useful for some architecture" and "Little conceptual models are nicely expressed." Another informant articulated a principle for choosing the focus: "80/20 rule: express that key part of the system that gives context for everything else." Another explained that he uses it only for "a very-high-level view of the classes and the flow", because otherwise "it clutters the information; it complicates the view".

Many expressed the principle of using UML as long as it was useful – and only that long: "I do as much as the problem demands." Issues of synchronization and consistency arose in this discussion, and focused use was described as a way of avoiding such issues: "Relations between representations? Inconsistency between different views? Interactions between abstraction layer and detailed layers? I focus on functions and create one possible candidate of the class diagram…" and "I don't elaborate if a high-level description is sufficient for decision-making."

The interest is in providing traction for solving problems or making decisions without incurring undue costs: "I like to help people do things better than they do now, without big overheads." There were repeated references to cost-benefit balance: "To try to use rigor on principle to make things better just adds cost; better to use what works and to use rigor when you really need it." Similarly: "'Meanwhile, in the real world', it's just too much work to go the whole hog all the time." Another informant, who works for a company whose software tools support UML, offered a secondary report, that consumers use the UML tools his company produces selectively "as a means to get a start point in the code" and don't persist in using it over time.

F. Which Parts of UML Were Selected?

Which parts of UML these informants use depends on the problems they are addressing (e.g., those who focus on networks tend to use sequence diagrams) and on whether they are using it to assist their own thinking or to facilitate discussions with stakeholders (e.g., the only informants who report using state machine diagrams are those who use them to assist their own thinking only). Informants whose declared use is selective were asked which elements of UML they use. Table 2 summarizes their responses.

TABLE II. ELEMENTS OF UML USED BY THE 11 'SELECTIVE' USERS.

UML diagrams	**Number of users**	**Reported to be used for…**
Class diagrams	7	structure, conceptual models, concept analysis of domain, architecture, interfaces
Sequence diagrams	6	requirements elicitation, eliciting behaviors, instantiation history
Activity diagrams	6	modeling concurrency, eliciting useful behaviors, ordering processes
State machine diagrams	3	
Use case diagrams	1	represent requirements

The UML elements identified in secondary reports were the same as those in the table, with class, activity and sequence diagrams identified most frequently.

Some informants also specified elements of UML that they never use: state machines, use case diagrams. Some elements of UML were never mentioned: communication / collaboration diagrams.

G. What's a Use Case?

As one informant expressed it: "It's hard to design without something that you could describe as a use case." Most of the informants who use UML selectively mentioned use cases, but only one found use case diagrams to be of use. The informants were careful to draw a distinction between use cases and use case diagrams when describing their practice. Many described use cases informally, for example, as: "Structure plus pithy bits of text to describe a functional requirement. Used to communicate with stakeholders." Others described use cases as user stories, scenarios, narratives. In contrast, one regular, long-term selective user was explicit that he never uses use case diagrams, because they are "totally useless". Another "…tried to teach use cases at a company – couldn't. They roadblocked on the implementation details. Black box use case specification is hard for people."

H. Contexts, Tools, Other Representations

Selective users generate their UML both manually (via pencil-and-paper, post-its, whiteboards) and within UML tools (e.g., Rational Software Architect, Astah, MagicDraw, Eclipse plug-ins, in-house tools). Some keep it simple: "Whiteboard and digital camera are all I need – for 95% of usage." Some combine UML tools with wikis and other systems. Many integrate UML with other representations: "Not just UML: DFDs and other things – [my use is] goal oriented."

A variety of alternatives were identified as representations these informants use: algebras ("more concise, faster to think on paper"), ADS (described as "UML on steroids"), flow charts, block diagrams, entity-relationship (ER) diagrams, SBVR and BPMN. One informant prefers "GML: galactic modeling language – boxes and arrows". Predecessors (from past use) were also identified, including CoRE, SSADM and Booch notation.

IX. Wholehearted Use

Although none of the informants declared this as their current use, there were secondary reports of organization-wide introduction of UML, with commitment from management and associated investment in champions, tools and culture change, with the intention that UML be used throughout the software development process. Two of the informants came from companies that invested in wholehearted, top-down commitment to UML in the past; in neither case had UML use persisted in that model, although pockets of UML use (both selective use and wholehearted use by particular groups) persist in both organizations. Other informants provided secondary reports about other groups or divisions in their organizations that attempted wholehearted adoption of UML. These secondary reports had recurrent features in common:

A. Investment in Examples, Tools and Education

Wholehearted use of UML is characterized by an organizational commitment to a change of culture and practice. Investment is made in tools and education, and 'champions' or visible early adopters are influential, because they provide authentic examples of relevance to the company, they help to develop and promote conventions (e.g. naming conventions) that assist communication, and because they are available for advice. For example: "… there were also a pretty large number of UML zealots in the services divisions at the time ... I think mostly because there was much better support for UML: education materials, tools, etc. Also, UML did evolve in the whole system direction, so that helped more people see their way to clear to adopting it."

B. Not 'Strict' But Adaptive Use

Even those in companies that set out to adopt wholehearted use found themselves adapting elements of the notations and developing tools or tool extensions. One informant described a merger of class and activity diagrams in order to facilitate discussions in the context of business processes. Another reported that his group developed management reporting tools for use cases which weren't in the software engineering tools: "We needed to compromise between technical and management views." Several remarked on in-house developments, either tools or conventions, that allowed informal annotations of diagrams, in order to assist dialogue and help with coordination between perspectives and views.

C. UML Was Not a Panacea

Using UML did not necessarily lead to success. Many of the secondary reports of wholehearted use were bound together with reports of projects that did not reach the market, despite the investment, or projects that did not satisfy clients, who found the UML representations (e.g., requirements, design proposals) complex and difficult to encompass, and therefore remained unconvinced that their needs were satisfied. One informant described producing a "massive UML diagram of their whole software structure" that encompassed pages of diagrams, accompanied by thousands of words of explanation in an associated wiki, and took several days to present. He expressed doubt that the documentation would be maintained as the system evolved.

X. Discussion

A number of pervasive themes emerged from the interviews, many of which resonate with the findings of other studies. UML use is by no means universal. For example, Aranda [3] reported results consistent with this study; he interviewed ~100 software developers, consultants, and entrepreneurs from 15 relatively small organizations, as well as Microsoft and IBM employees. None of them practiced model-driven development. A single firm among those studied used UML to some significant extent, and Aranda also spent three weeks conducting observations there (reported as "Bespoker" in [4]). Aranda concluded that, in the case of this firm, the data suggest that UML is used primarily for contractual and

business relations reasons, by analysts, and is largely ignored in the rest of the development process.

Even among those who adopt model-driven development, UML is not yet universally accepted as the modeling language of choice. Hutchinson et al. [13, 14] found, in their study of model-driven engineering in industry, that people tend to use multiple modeling languages. Companies using MDE tend to be building on domain-specific languages (DSLs), and their notion of DSL is very product/implementation focused.

Evidence from this study suggests that UML is used in both 'greenfield' and 'brownfield' contexts. For example, one informant described how UML representations of existing software were used in collaborative discussions about integration. Hutchinson et al. (ibid.) found that productivity gains from code generation tended to outweigh losses from integration with existing code – but the implementation-focused use they observed was associated with a risk that MDE may prevent organizations from responding to new business opportunities.

A. Not by Me, But by Someone Else

There was a tendency for informants in large organizations who did not themselves use UML, or who used it selectively, to assume that colleagues in other roles were likely to use it more. One informant, a software architect who declared selective use, suggested that UML plays a greater role in implementation: "Generating structured code from a UML model – is gaining increasing use." Yet another informant, a developer, offered a contrasting view: "Usually used at the front end – then we transfer [attention] to the code itself." Many could envisage more intensive UML usage, for example: "I can imagine that there are times for rigorous UML…very software intensive… safety or security applications may want that sort of rigor." Across the sample, analysts pointed to software architects as more likely users, software architects pointed to implementors/developers, developers pointed back to architects or to developers of other types of software, and so on. There was no one 'role' that claimed UML particularly.

However, the belief that 'someone else' used UML was not universal; many informants observed explicitly that colleagues who might be expected to use UML did not, e.g.: "My experience is that the majority of the developers and software architects (interestingly) do not use UML. Some of the software architects use BPMN." In another example: "Most of my work over the last few years has been on the side of understanding the domain, and thinking about what users need to support their thinking and work. Then other folks on the team do the design/architecture and implementation/development of the tool software. That having been said, even they didn't do much UML."

B. Support for Communication as a 'Lingua Franca'

The observation by one informant (see 'selective use', above) that UML can be particularly helpful in international collaborations, because it provides a bridge to help participants with lower spoken language proficiency discuss ideas and resolve ambiguity, puts a different perspective on the 'lingua franca' label. In this case, UML is truly being used as a 'language of exchange' between people of different mother tongues. Arguably, we need to look more closely at the differences between using UML to *communicate* models (and to overcome differences in perspective in reaching a common understanding of those models), and using UML for collaborative discussion in order to *build* models. The key to collaborative dialogues is discussion over a shared representation that either means the same thing to all parties or enables them to identify and correct differences of understanding – and then re-represent in a way that's more accurate. For some, UML serves this purpose. However, for others it does not: "The translation costs of bridging across different people and different levels and different ways of thinking – are just too high." In another example: "I started using UML 15 years ago, and to be honest I've been using it less and less. It can be very hard to communicate with people, except in a very simplified way."

C. Early and Late – But Not by the Same People

Responses concerning UML use tend to be polarized, between design use and implementation use (cf. [9]). Fowler [10] distinguishes three types of UML use: sketch, blueprint, programming language. Responses suggest selection among these, rather than transition. As articulated by one informant: "Both ends of the spectrum – but not by the same people." Another observed of his own organization: "Most UML use is implementation, but much of early design is also UML". Despite the notional accommodation of the whole process, informants tend to use UML either in early design, or in implementation, rarely both (even when informants' roles include the whole process).

D. Fluid Enough for Design vs. Precise Enough for Detail

Several informants noted the conflict between what they want from a modeling notation – useful abstraction – and what they need for implementation – precise formalism. Lange et al. [17] remark on this conflict: "The generality and freedom that enable UML to cater to this wide range of purposes are also the source of its weakness. UML has no formal semantics. This poses a problem when different people use a UML model; and because one of UML's main purposes is to communicate about a design, different ways of using UML are potential causes of communication problems." (p. 43)

Those who use UML during early design tend to use it selectively and informally. Lange et al. [ibid.] also found that: "…adherence to the [UML] specification is rather loose. This might be a result of UML's lack of formal semantics and large degree of freedom in its application. On the other hand, it might be that informal use is "good enough" for many practitioners' purposes" (p. 41). This resonates with the informant's explanations, e.g.: " I guess that's because … it was felt that the software was small, and being completely re-done regularly, and … the developers were 'high end' and could keep the design in their head effectively and communicate effectively without formal diagrams etc." Those who use UML as a thought tool want flexibility. "You have to follow the rules or you can't use the tool. But I'm always cavalier with the rules."

Those who use UML in design tend to use it to express models, abstractions. Their use stops when then think more concretely and specifically about implementations, e.g.: "Usually used at the front end – then transfer to the code itself." Comprehensive UML diagrams of large systems are reported as complex and unwieldy by those who have experience of wholehearted use. "The latest Rational tool is so bloated that it screams 'Don't use me!'" Informants also express concern about the "clutter" and complexity of UML, with consequent issues of completeness, consistency, and synchronization (cf. [12]). Informants explain that using UML selectively keeps the complexity under control and avoids issues about completeness (because completeness is not a goal of their use) and consistency across representations (because they address consistency within their selection and use of a UML subset for a focused purpose). They find better support for analysis of implementations in tools associated with program code. Similarly: "These explanations support the argument that the UML may be too complex. … Focusing on a smaller set of components … may be a better strategy for both analysis and students in the early stages of using UML, and may reduce the cost of ensuring consistency across different components." ([7], p. 112)

E. What Is Useful About UML?

The utility of UML for the practitioners in this study rests in its fitness to address their purposes. The informants identified as useful representations that, as they use them:

- are understandable and fit for purpose (that is, for the practitioners' purpose, not one imposed by methodology or ideology);
- capture structure at the right level (e.g., class diagrams are considered useful; object and use case diagrams are not identified as useful);
- make behaviour explicit (e.g., by showing a sequence of actions).

Informants criticized UML for its complexity, lack of formal semantics, inconsistency, and issues of synchronization between different diagrams (and between models and code). This resonates with [8], p. 83: "…the human thought process in system architecture, analysis, and design involves the constant interplay between a system's structure and its behavior. However, rather than supporting this thought process, UML's separation of the system model into different views, represented by different diagram types, dictates and enforces the damaging segregation of structure and behavior, thereby obscuring the developer's overall system comprehension."

F. The Lingua Franca of SE Education?

Professional practitioners on the whole are simply not thoughtless, ignorant, bigoted or stupid. There is an implicit assumption in the academic discourse that declining to use a specific methodology or formalism equates to declining methodology or methodical practice. On the contrary, evidence from empirical studies of professional software developers (e.g., [20]) shows clearly that professional software developers who decline specific tools nevertheless demonstrate that they have thoughtful, systematic practices. Many practitioners already have a repertoire of tools and representations that have been thoughtfully developed and evolved over time to fit their effective practices. There is a need to design tools that relate to existing needs and practices, rather than dictating costly change. One of the major reasons for declining to use UML is, as one informant phrased it: "UML is not just a notation, it's an ideology."

On the other hand, students typically don't already have such a repertoire, and they may benefit from exposure to useful concepts, methods, and notations. Even an informal survey of current textbooks and course syllabi shows that UML has achieved penetration in software engineering education. One of the informants pointed out: "Based on my experience in IT, I came to the conclusion that UML is primarily an academic thing." Several informants perceived that UML is a useful credential, even if it is not used in practice: "A friend of mine says that it's essential to learn UML to land a job as a business analyst, even though you'll probably not use it once you got your job, by the way." One might conjecture that UML is effective in software engineering education, because of what it captures, and where it directs attention, rather than as a prescription for design actions. This is a matter for further research.

XI. Threats to Validity

The work reported here is indicative, rather than definitive. The sample is large enough to give confidence that it represents significant views; it is not large enough to claim comprehensiveness. On one hand, the study characterizes authentic use (or non-use); on the other, there may well be other categories of use that were not represented in this sample. We attempted to mitigate this risk by intentionally sampling a broad range of organizations, small and large, across a broad range of industries, and a broad range of age of companies – and by limiting the sample to one informant per company.

The study relies on self-report. We mitigated this risk in two ways. First, some interviews were augmented with observation when opportunity arose, which allowed independent verification of the informants' reports. Second, informants were asked to provide actual examples of their UML (in confidence). The observations and examples confirm the responses and instill confidence that self-reporting in this particular study is appropriate and reliable. Previous studies (e.g., [20]) that combined interview with observation (allowing independent verification of informants' accounts) also suggests that experienced professionals are reliable in their accounts of their practice.

The categories of use emerged from the data-driven analysis conducted by the author. To mitigate the risk of researcher bias, the analysis was subjected to a limited form of validation: expert review by two independent, experienced, professional software developers. Each reviewer was presented with the categories and with a representative selection of anonymized data, and each was asked both to assess the adequacy of the categories for characterizing the data and to place the examples into categories. The reviewers agreed that the categories reflected the data appropriately –

with one exception: one reviewer considered that the 'retrofit' category did not merit a separate heading and suggested that it should be included within 'no UML'. The reviewers categorized the examples with complete agreement both with each other's categorization and with the original categorization, i.e., inter-coder reliability was 100%.

XII. Conclusion

This empirical study complements and resonates with other studies of UML use in industry, finding (as others do) that practitioners take a broad view of what constitutes 'modeling', and that, even if UML is viewed as the 'de facto' standard, it is by no means universally adopted. The majority of those interviewed simply do not use UML, and those who do use it tend to do so selectively and often informally.

What the study adds to the discourse is clear evidence of different patterns of use, expressed in the voices of the software developers who were interviewed, in the context of genuine professional practice. The observations are informed by a sample that represents the software development community, rather than just the UML user community. Even within these different patterns of use, there are a number of issues that challenge the effectiveness of UML as a lingua franca – but there are also practices that employ UML effectively in reasoning about and communicating about design, both individually and in collaborative dialogues.

The different patterns imply different purposes and needs – and hence different implications for tool support. They also highlight some of the fundamental tensions within UML, resonating with arguments [8], [25], [17] that UML's intended strengths (i.e., generality, accommodating different levels of abstraction) are intimately associated with its observed weaknesses (e.g., latent complexity, issues of transformation and coordination between views) and arise from fundamental properties of UML (e.g. lack of formal semantics, separation of expressions of structure and behavior).

The study highlights the need to consider the relationship of tools, including notation, to both the community of practice and to the domain of application. It makes clear that software developers are open to useful concepts and tools, but will not adopt tools and ideologies at odds with their considered practice.

Acknowledgments

Thanks to the professional software developers who shared their experience and examples. Thanks also to: Jorge Aranda, David Bowers, David Budgen, Andre van der Hoek, Shailey Minocha, Dave Roberts, Kevin Waugh, Jon Whittle. This research has been supported by a Royal Society Wolfson Research Merit Award.

References

[1] Anda, B., Hansen, K., Gullesen, I., and Thorsen, H.K. (2006) Experiences from introducing UML-based development in a large safety-critical project. Empirical Software Engineering, 11, 555-581.

[2] Andersson, H., Herzog, E., Johansson, G., and Johansson, O. (2010) Experience from introducing Unified Modeling Language/Systems Modeling Language at Saab Aerosystems. Systems Engineering, 13 (4), 369-380.

[3] Aranda, J. (2010) A Theory of Shared Understanding for Software Organizations. PhD thesis, University of Toronto.

[4] Aranda, J., Easterbrook, S., and Wilson, G. (2007) Requirements in the wild: How small companies do it. 15th IEEE International Requirements Engineering Conference (RE'07), 39-48.

[5] Booch, G. (1994) Object-oriented analysis and design with applications, second edition. Redwood City: Benjamin/Cummings.

[6] Budgen, D., Burn, A.J., Brereton, O.P., Kitchenham, B.A., and Pretorius, R. (2011) Empirical evidence about the UML: a systematic literature review. Software: Practice and Experience, 41 (4), 363–392.

[7] Dobing, B., Parsons, J. (2006) How UML is used. CACM 49, 109-113.

[8] Dori, D. (2002) Why significant UML change is unlikely. CACM, 45 (11), 82-85.

[9] Forward, A., and Lethbridge, T.C. (2008) Problems and opportunities for model-centric versus code-centric software development: a survey of software professionals. Models in Software Engineering workshop (MiSE '08) at ICSE, ACM, 27-32.

[10] Fowler, M. (2003) UML Distilled Third Edition: A Brief Guide to the Standard Object Modelling Language. Addison-Wesley, 2003.

[11] Glinz, M. (2000) Problems and deficiencies of UML as a requirements specification language. Tenth International Workshop on Software Specification and Design. 11-22.

[12] Grossman, M., Aronson, J.E., and McCarthy, R.V. (2005) Does UML make the grade? Insights from the software development community. Information and Software Technology 47, 383-397.

[13] Hutchinson, J., Whittle, J., Rouncefield, M., and Kristoffersen, S. (2011a) Empirical assessment of MDE in industry. ICSE'11. 471-480.

[14] Hutchinson, J., Rouncefield, M., and Whittle, J. (2011b) Model-driven engineering practices in industry. ICSE'11. 633-642.

[15] Jacobson, I., Jonsson, P., and Overgaard, G. (1992) Obect-Oriented Software Engineering: A Use Case Driven Approach. Reading: ACM Press / Addison-Wesley.

[16] Kobryn, C. (2002) Will UML 2.0 be agile or awkward? CACM, 45 (1), 107-110.

[17] Lange, C.F.J., Chaudron, M.R.V., and Muskins, J. (2006) In practice: UML software architecture and design description. IEEE Software, March/April 2006, 40 -46.

[18] Nugroho, A., and Chaudron, M.R.V. (2007) A survey of the practice of design – code correspondence amongst professional software engineers. International Symposium on Empirical Software Engineering and Measurement (ESEM '07), ACM, 467-469.

[19] Nugroho, A., and Chaudron, M.R.V. (2008) A survey into the rigor of UML use and its perceived impact on quality and productivity. International Symposium on Empirical Software Engineering and Measurement (ESEM '08), ACM, 90-99.

[20] Petre, M. (2009) Insights from expert software design practice. ESEC/FSE'09, ACM, 233-242.

[21] PR9.NET (2005) Wide gap amongst developers' perception of the importance of UML tools. Press release, 20 April 2005. http://www.pr9.net/comp/development/1674april.html

[22] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy,F., Lorenson, W. (1991) Object-Oriented Modeling and Design. Englewood Cliffs: Prentice-Hall.

[23] Selic, B., Kent, S. and Evans, A. (2000) UML 2000 – Advancing the Standard. Springer-Verlag, LNCS number 1939.

[24] Störrle, H. (2001) Describing process patterns with UML. Software Process Technology, 2077/2001, 173-181.

[25] Thomas, D. (2004) MDA: Revenge of the modelers or UML utopia? IEEE Software, May/June 2004, 22-24.

[26] Tichy, W. (2011) Empirical software research: an interview with Dag Sjøberg, University of Oslo, Norway. ACM Ubiquity, June 2011, 1-14.

[27] Zeichek, A. (2002) Modelling usage low: developers confused about UML 2.0, MDA. SD Times, 15 July 2002. http://www.sdtimes.com/link/26637