**Globally Speaking**

**Podcast 86 Transcript**

**All About Internationalization**

**Featuring Jeff Knorr of International Labs**

Michael I’m Michael Stevens.

Renato I’m Renato Beninatto.

Michael And today on Globally Speaking, we are getting schooled on the topic of internationalization.

Renato Internationalization is one of those things that everybody talks about, but very few people know really how to do it and how to deploy it. So, our guest is an expert, has been around from the beginning of internationalization that was born at the same time as localization.

Michael It was a really fun conversation and I think our listeners are going to learn a lot.

Renato So, let our guest introduce himself.

Jeff I’m Jeff Knorr. I’m the Managing Director of Internationalization Labs based out of Boulder, Colorado, and I’ve been in the globalization business for over 25 years.

Renato It’s a long time. Were you one of the founders of this industry?

Michael [Laughs] In 25 years?

Jeff Yeah, it’s a long time, I think I started out helping globalize DOS programs and Windows 3.1, you know, early stuff, driver software, but internationalization wasn’t even really a coined term back then. We sort of did internationalization in real time with localization.

Renato How would you define internationalization for our listeners who don’t know what it is?

Jeff Well, I’ll define it as it applies to software globalization, okay? Software globalization is the sum of internationalization and localization. Internationalization is really taking a piece of software and making it prepared for localization independent of any one locale or language. Typically, if you don’t internationalize first, localization may be impossible. So, if you just start translating a piece of software, you’re likely to break functionality, the translation won’t appear properly in the interface and you’re not going to get a product that looks like it was developed for its intended locale or audience.

Michael Jeff, some people might say, you know, haven’t we overcome that with some of the more recent programming languages? Like, if you’re doing a mobile app, the strings file contains all this stuff, so what’s the big deal? When you’re writing in Ruby or Python, don’t these things naturally solve themselves?

Jeff No, [laughter] you know, if they did, I wouldn’t be in business; I’d probably change careers a long time ago. I’ve been hearing that ever since the onset of Java, when Java hit the market. Essentially, what’s happened is the most modern programming technologies allow for standards to create easily globally ready software.

 The problem, I think, for lack of a better word, is globalization ignorance, really. Not to be too harsh, but, you know, our universities and courses for producing programmers—globalization is not really a hot topic. How most software gets developed is fly by the seat of your pants; you get this really great idea and you start just programming. And it’s usually, you’re just targeting your local audience in your language and locale standards, and then you produce this piece of software completely ignorant that, oh, well, in Java or dot.net or Ruby or anything, there are standards for how you want to, for example, resource strings so that they can be easily localized, or how you want to make sure that you’re identifying the proper locale and therefore using the necessary internationalization standards for that locale and language.

 So, unless you harness those innate globalization functionalities in each technology, you can easily create a program in any technology that’s not globally ready.

Michael So, what I’m hearing from you is good coders may just neglect that element of good code.

Jeff From a globalization perspective. I’ve had the privilege of being exposed to brilliant programmers, and the things that they’ve been able to do and architect with software is just mind-blowing. And then I’ll come on the scene, or we’ll bring in consultants on the scene, and introduce them to the wonderful world of now building that code for a global audience and thinking about language constructs. There’s a lot that we can bring to the table to educate them on how they can take their software to the next level and introduce it to a global audience.

Renato So, in the old days, internationalization had a lot to do with sizing of boxes—you mentioned Windows 3.1—and you had those boxes and then you had text in German that was too long, and you had to try and change coordinates to make that stuff happen. And then you had double-byte languages like Chinese and Japanese and you needed to make sure that the resource files and the strings would not break and have weird characters appearing in the middle.

 Is it still the same or has it moved on to more complex things? One of the things that didn’t exist when you started in the industry 25 years ago was the widespread use of Unicode. Did that have any impact in internationalization? Because I know that many people think, “Oh, it’s Unicode, so we don’t need to internationalize.” I’m sure you’ve heard that before.

Jeff [Laughs] Yeah, yeah, I think there’s just a big misunderstanding of what Unicode is. And you’ve hit on a bunch of topics there, right? Okay, so, everything from string resourcing to UI design, and certainly encodings and how to handle that—it’s the same internationalization best practices that need to be applied regardless of the latest technologies that you’re developing with. The great thing is that new technologies, programming technologies that come out are taking this into consideration. It’s just that the architect or programmer building these programs in those technologies have to account for them and implement them appropriately based on those standards.

 Unicode, we’ll take that as an example. That came on the scene in the mid-nineties for the very reason that there used to be a series of encodings based on the language. I mean, encoding is just essentially code points or numbers being assigned to every character in a character set. Okay? So, ASCII, for example, was one of your first encodings. It didn’t account for any extended characters, anything even in Western European, let alone even all of the characters to support English. So, then you had all these different encodings you had to support, and it was a virtual nightmare. And, you know, early adopters of globalization like Microsoft and Apple and such, they made functionality to allow for the conversion between encodings and such happen, but it was still just a virtual nightmare.

 Along comes Unicode. And Unicode was just basically founded on the premise of creating essentially a universal character set, first and foremost, so it includes nearly all of the characters, symbols, punctuation marks, everything, of all the scripts used throughout the world by humans, right, and then a series of encodings, namely you’ve heard UTF-8 or UTF-16, that basically assign codes for each character.

 What this allows you to do is not have to convert between all these different native encodings. To truly have a globalized software application, it would make the most sense to really be adopting Unicode, the Unicode encoding system, from the very beginning. Sometimes that’s not always possible, primarily because some applications may need to be backwards-compatible with some older technology or they may have an external data feed that is not Unicode-compliant.

 But, you can still deal with this better—at least within the application, everything is Unicode. At certain boundaries in this architecture, you can have conversions or trans-coding that happens. But Unicode makes things much easier, especially when you’re dealing with things like multilingual databases where you can have 10, 20, 40, 50 languages all stored in a single database that’s Unicode-enabled.

 With older programming technologies, it was much more difficult to back up and talk about all these internationalization concepts. It was more difficult to internationalization-enable a C or C++ program than, say, than a dot.net program or Java program. It’s become easier and easier and there’s been a lot of standards in place that allows for one to create a truly globalized piece of software. It’s just a matter of being aware of those standards.

 That’s what I try to do with our company. I pride ourselves on being in the education business, if anything else. We teach companies how to produce globally ready software, first and foremost. And whether we do that with code assessments, with training, consulting or actually partnering with them to enable their own code, the whole point is to make them truly global software providers and make sure they get these best practices and can maintain their software moving forward.

Renato Let’s bring this to something very practical. I actually remember a story that happened a few years ago when Foursquare were still a big thing, and they launched in many countries and there was some internationalization problem with the code, and it was broken everywhere. What kind of problems do you find when the software is not internationalized?

Jeff There’s a number of problems. Let’s name some of the most common. First and foremost, probably with every single internationalization project I’ve ever been involved with, is the problem of hard-coded strings. And this is where you have string literals or content—whether they’re error messages, whether they’re the text in the user interface—that need to be translated, is buried in the functional code. You have to isolate or externalize those strings into resource files so that they can be safely and properly localized. If you don’t, what you’re essentially creating is different language versions of your source code, and that is the antithesis to internationalization. You need to isolate the functional code from the language- or locale-sensitive aspects of it.

 There’s always string externalization issues. It’s probably one of the largest initiatives in any enablement, but it’s probably the least complex. Some of the more complex are things like Unicode migrations. So, a piece of software may be using native encodings and haven’t adopted Unicode, and yet they’re trying to produce some kind of a multilingual solution.

 So, that gets difficult because now you have to look out for all of the string-handling functionalities and programming methods that are making assumptions based on a specific character set or native encoding, which isn’t going to work—usually they’re probably parsing strings, counting bytes, and it’s going to break.

 International formatting—date, time, number, currency, all of that—those are pretty typical. Locale handling or the ability to switch locales or identify the locale of the user and be able to provide them with the correct localized content.

Renato I can think of concatenation. This specific problem of Foursquare was that the sentence was written in English with variables, and then when you read it in another language, the sequence of words is different than it is in English or there is no pronoun or the pronoun changes, and the concatenation becomes horrible in the translation and creates sentences that don’t make any sense. That’s a very common issue also.

Jeff Absolutely. Concatenations are taboo; it’s a dirty word in internationalization. We flush those out, you know. Another one is trying to append an ‘s’ at the end of something that’s pluralized.

 You get in other problems in software interfaces: leveraging strings from multiple areas in the interface. So, as you know, if you take one noun and you start moving it around to different areas, that’s going to be a problem as far as the different contexts where you have languages where you assign gender and such. There’s always some type of expansion once you translate, and you want your localized interface to look like it was produced for that language.

 Like you were saying earlier, German being very long in horizontal length, Spanish being short, but then you also have vertical expansion when you’re talking about Asian languages, you know. You might have to increase the point size because if you’re using a small one, your Chinese characters are going to look like ink splotches. So, there’s a lot of design considerations you have to look at, too. You want the interface to look like it was designed around the language the user is viewing, so there’s a lot of those considerations.

Michael Is there an industry that suffers the most from these challenges?

Jeff Everybody’s affected by this now, especially that there’s an app for everything now, right? Everyone who used to have a desktop application is now moving to web-based applications and now mobile applications, and the headache with this is that they’re very easily deployable. Somebody in development gets a request from somebody in marketing or sales who say, “Hey, I have this customer, they’re dying to have this in Japanese.”

 Well, you can easily just get it deployed but, you know, there’s a lot of work to do it. It’s easy to deploy a piece of software these days globally, but that just means you have to go quickly to market to get it internationalized and localized. Technology is moving so fast these days. It’s hard for us to even sometimes keep up with the latest, greatest technologies and be able to support a customer that comes to us who’s developing EMOs [embedded mobile operating systems] and be able to tell them, “Okay, here are the internationalization standards; this is how you need to adjust this for your target locales.”

 But I don’t think this is specific to any one industry. We’ve worked from healthcare, to IT to small to Fortune 100 companies. This is in every corner of software development.

Renato Two internationalization engineers walk into a bar. What are they going to talk about?

Jeff They might talk about some of the interesting software they’ve had to support Arabic, for example. Arabic and bidirectional languages have gotten very popular. And that is confounding to people that are new to globalization. Just the whole aspect of displaying an interface right-justified and, you know, to be able to adapt your software to that, I think it’s almost like you’re looking at something foreign, not from this planet.

Renato Yeah, because it’s actually foreign, right!

[Laughter]

Jeff It is foreign! You’re talking about a whole interface flip!

Renato Let’s give some enlightenment here for the people who are not familiar with the concept of bidirectional languages, right? It’s mostly Arabic and Hebrew, and why do we call them bidirectional?

Jeff For example, you might read a sentence in Arabic that says, “If you have any problems, please contact Microsoft Support.” Well, you’re going to read this whole sentence from the right to the left, but when you hit “Microsoft,” you’re reading it left to right, and then you continue on reading right to left.

Renato So, it could be an email, it could be a number, right?

Jeff Yeah. They call it bidirectionally because there are some terms that are simply either not translated in Arabic or they’re proper terms and you would read those left to right. And, of course, the interfaces are all going to be right-justified as opposed to typically left-justified, which is the standard in every other set of languages.

 In the medical and life sciences arena, it’s growing big. There’s a lot of globalization happening there. Most people are not just releasing, say, desktop applications; usually web applications and mobile applications. A lot of folks are trying to migrate to more modern technologies. I mean, everyone’s accessing things by their mobile devices now, so we’re riding that wave, and it’s just as long as folks are coming out of colleges with their computer science degrees and they’ve not been exposed to any type of programming for the global world, we’ll be here to educate them.

Michael To do that.

Renato There’s this evergreen effect, right?

Jeff Right.

Renato There’s always a new generation that hasn’t lived through this. The topic *du jour* is AI and machine learning. How does that help or affect internationalization, if at all?

Jeff So far, I would say it hasn’t affected it much, if any at all. It’s going to hit, I think, the localization end or the translation side, certainly it already is. I envision a world—and it’s not going to be, it’ll probably be within our lifetime—where you’re going to see the ability to globalize a piece of software with AI technology probably at the same time that programs start programming. That’s the scary future.

 Eventually, you’re going to have programs that can program more efficiently, with less errors, that will be bug-free right off the bat. You can, you know, feed it in the right requirements or configuration into its algorithm and it will automatically be globally ready, just translated, and maybe that’s done with machine translation. That will be the end of a lot of jobs—it won’t be just internationalization. I think developers will be scrambling then, when that happens.

Michael There’s a lot of talk about what it could achieve, but the real sort of feet on the ground is not quite there and pretty far away at this point.

Jeff I think it could grow exponentially. There’s a big race, it’s almost as like an arms race, really, with the big powers of the world trying to be the first to come up with the AI technology that’s truly going to rule them all. I think you’ll see it applied to software development and even maybe before software development, you’ll see it applied to testing and QA. I think that’s an area where it would really benefit.

Renato Let’s say the owner of a translation company has a client in Moldova and they want to write software that is internationalized. How do they go about that?

Jeff I say, if you knew from the get go you wanted to develop a product that was ready to hit the Asian and European markets right away, know what those languages are, know that you’re adopting Unicode and technologies that support those languages, and depending on the technologies that you select to build your architecture from, know the standards. There’s a wealth of information on the internet, or you can come to an internationalization service provider that can produce training. There’s certainly training that can happen to get all of your testers, your developers, your product managers up to speed on internationalization best practices so you do it right the first time. And you can have consultants hold their hand along the way as they build it out.

 If you have a pre-existing product and this company is coming to you and they say, “hey, we need this translated in these 20 languages,” and your solution architects try to scope this for localization and say, “hey, this isn’t even internationalized yet; you can’t localize this yet,” in today’s day and age, there’s not a lot of in-house internationalization teams in the largest localization service providers. I used to work for one of them. They didn’t have internationalization teams for the only reason that internationalization expertise doesn’t grow on trees. There’s a huge demand for internationalization experts. Just dabbling in internationalization for a year or two on a software product doesn’t make you a consultant that can leverage that and teach others. There’s a whole world out there, and depending on the technologies, it could be completely different how you approach it.

 That’s not to say that there aren’t folks within these localization service providers that have skilled localization engineers that have dabbled with internationalization. They know how to identify an internationalization issue, would roughly know what needs to happen, but they aren’t developers and they don’t have the skills to actually implement or maybe even confidently consult the customer on that.

 And that’s where companies like mine come into play, where we fill that gap, because managing an internationalization team is much different than managing a localization team. They’re a different breed; it’s really like a team of developers and consultants as opposed to what localization engineers might do.

 I started off as a localization engineer. Back then, we almost had to do internationalization on the fly because there was no internationalization team. And that’s when we…you know, certainly around the dot.com era is when internationalization just went wild. Everybody and their mother was trying to get on the global web, and they had no idea how to get supportive of Japanese or Chinese or Russian or what have you.

 We had to create a team very quickly. There’s no school where you get a degree in software internationalization. So, a lot of the ways that we produced internationalization consultants was we had a few that had been doing this for a long time. Some of these people were involved with the Unicode consortium, founders of that, and very involved in that community, or had been ad-hoc internationalizing programs for a long time.

 And what we’d do is we take localization engineers that wanted to further their career and get really deep down in it and we’d have them mentor under these internationalization engineers until eventually, they became internationalization consultants themselves. But, really, the only way you become a true internationalization expert is by doing it. And you can take any amount of training, but until you really get your hands dirty in it…

I work with, you know, very skilled internationalization engineers that are able to interpret someone else’s code and be able to flush out the internationalization issues, come up with solutions to those issues and educate the customer on how to maintain an internationalized product, best practices and standards.

 It’s not every day that a company is taking their software global for the first time. That’s when we get pulled in, right? I have always dubbed the motto that you “internationalize once, localize many.” The premise being that once you internationalize, you better get it and make sure your teams learn it so that you can maintain it moving forwards and you’re not going to have to constantly re-internationalize. You internationalize it, you maintain that internationalization requirement and then you add languages and improve upon that translation or grow it as your product grows.

 So, we don’t get a lot of repeat business unless we’re working for a behemoth like Google who’s constantly acquiring new technology that needs to come into the globalization fold. We’re constantly looking for that next new product that’s going to go global for the first time. That’s our market space.

 That’s difficult to manage for a localization company. Localization companies are usually dealing with volumes of content that either grows because the product is expanding in new versions, or you’re talking about very dynamic content, or they’re adding languages, they’re tapping new foreign markets. And so, those localization accounts are sort of like the gift that keep on giving. That doesn’t happen with internationalization, necessarily.

Michael And the localization companies don’t necessarily have the experience or in-house expertise to identify the internationalization issues and start working to resolve them. Hence, why they reach out to you to support them.

Jeff Right, and, right, and usually, the most skilled folks in any of these localization providers are within their solutions teams. These are the veterans, these are the guys who come up with the solutions to complex localization jobs—what CMS to implement, machine translation if necessary or possible, etc.—and they will be the first to raise the red flag and say, “Hold on, you’re not internationalized yet; let me introduce you to some folks.” And those are the people that I work with and have known, and they’ve been in the industry nearly as long as I have or longer. So, those people do exist, it’s just that there’s not likely to be internationalization production teams within those companies.

Michael Well, Jeff, my hope is after listening to this, our listeners now know some of the pitfalls and know what they don’t know and now know a place to go to resolve that.

Renato The mantra is internationalize before you localize.

Jeff Yes.

**End of conversation**