UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN LETRAS
FACULTAD DE FILOSOFÍA Y LETRAS
INSTITUTO DE INVESTIGACIONES FILOLÓGICAS

Hacia una semiótica de la creatividad en poesía digital

TESIS

QUE PARA OPTAR POR EL GRADO DE:

DOCTOR EN LETRAS

PRESENTA:
SEYED MOHSEN EMADI

TUTOR PRINCIPAL
DR. RODOLFO MATA SANDOVAL (IIFL-UNAM)

COMITÉ TUTORIAL
DRA. SUSANA GONZÁLEZ AKTORIES (FFYL-UNAM)
DRA. MARÍA ANDREA GIOVINE YÁÑEZ (IIB-UNAM)

CIUDAD DE MÉXICO, SEPTIEMBRE, 2022

Agradecimientos

Fue un largo viaje por un camino interior. El exilio lo alargó. Todo lo tangible se ha transformado en una realidad virtual más pesada que la física. Era imposible sin bondades, amistades, paciencia y disposición infatigable. Agradezco a mi madre por su amor y aliento. Ahora, han pasado años desde nuestro último abrazo. Ella resistió la pandemia para darme la promesa de otro reencuentro. Agradezco a mi tutor, Rodolfo Mata, su sabiduría, cuidado y paciencia durante este viaje, a Susana González Aktories por toda su ayuda, comentarios y los reencuentros supervisores. Agradezco a María Andrea Giovine, Raine Koskimaa, Giovanna Di Rosario, Shekufeh Mohammadi, Al-Dabi Olvera, Monica Maorenzic, Behnam Esfahbod, Philippe Olle Laprune, José Ángel Leyva y Antonio Gamoneda por su presencia, aliento y conocimientos.

Índice

Resumen

El desarrollo de la literatura digital como movimiento literario comenzó a finales de la década de los ochenta del siglo pasado. Sin embargo, el rastreo de la mayor parte de las ideas creativas de la poesía digital se puede remontar hasta la poesía de la primera mitad del siglo XX. E-Lit es una forma cultural emergente y tiene que ver con la transformación de las formas y los valores literarios a través de los medios de comunicación. La "creatividad" es uno de esos valores sujetos a cambios. El corpus documentado que existe de E-Poetry rara vez ha sido analizado a través del cuerpo textual íntegro de la poesía digital (texto literario, algoritmos, código, etcétera) y la arqueología de la puesta en práctica de las ideas creativas en la historia de la poesía moderna. Los límites borrosos entre lector y autor, donde "el medio se presta a la práctica experimental, sobre todo con las formas que alteran las nociones tradicionales de subjetividades estables y discursos centrados en el ego" [Hayles, 2007, p.17], invitan a hacer un profundo estudio comparativo entre la creatividad egocéntrica y la creatividad colectiva-interactiva. El modelo de sistemas de Mihaly Csikszentmihalyi y el enfoque sociocultural de Keith Sawyer proporcionan la plataforma básica para un estudio de esta naturaleza. Por lo tanto, esta tesis tiene como objetivo investigar el desarrollo de la concepción de la creatividad dentro de la poesía digital, no sólo a través de la arqueología de las formas y las estrategias creativas, sino mediante una revisión microscópica de los códigos fuente.

Introducción

Aunque se considera que la primera pieza de la literatura electrónica es el poema generativo de Theo Lutz de 1959 [Di Rosario, 2011, p. 68], la formación de la literatura digital o electrónica como movimiento literario comenzó a finales de los ochenta [Hayles, 2008, p. 6]. Sin embargo, la mayor parte de las ideas creativas de la poesía digital podría rastrearse en la poesía de la primera mitad del siglo XX [Funkhouser, 2007, p. 33]. E-Lit no es solamente la poesía, la ficción, el hipertexto, los juegos, el código de trabajo, o alguna nueva mezcla de todas estas prácticas, pero es -sin duda- una forma cultural emergente y tiene que ver con la transformación de las formas y los valores literarios a través de los medios de comunicación [Tabbi, 2007]. En particular, el término Poesía Digital:

> Applies to artistic projects that deal with the medial changes in language and language-based communication in computers and digital networks. Digital poetry thus refers to creative, experimental, playful and also critical language art involving programming, multimedia, animation, interactivity, and net communication. [Block, 2004, p. 13]

La poesía digital, ya sea considerada como una “migración” de la poesía impresa a los medios digitales (como lo considera Joseph Tabbi [Tabbi, 2007]) o vista como la “combinación de la tecnología y de la poesía” (como lo es para Jason Nelson [Nelson, 2011]), es capaz de deconstruir la antigua dualidad filosófica de esencia y apariencia, o de ser y devenir, y de desafiar las ideas románticas sobre la originalidad, la espontaneidad y el genio individual.

Un estudio adecuado sobre el desarrollo de la creatividad en la poesía digital debe considerar tanto la concepción y el desarrollo de la creatividad en la era de la impresión como las tecnologías para la aplicación de la creatividad en el mundo digital. Al respecto, Janet Murray indica: “giving the audience access to the raw materials of creation runs the risk of undermining the narrative experience” [Murray, 1998, p.40]. Sin embargo, luego parece contradecirse al agregar que: “calling attention to the process of creation can also enhance the narrative involvement by inviting readers/viewers to imagine themselves in the place of the creator.” [Murray, 1998, p.40]. Esta contradicción afecta incluso la labor de las organizaciones canónicas del campo, como la “Organización de Literatura Electrónica”, ELO por sus siglas en inglés, la cual no puso en línea la documentación del código fuente de los programas de poesía digital publicados en sus primeros dos volúmenes antológicos. El corpus documentado que existe de la poesía digital rara vez ha sido analizado mediante la aproximación a la variedad de los algoritmos, el código del programa y la arqueología de la puesta en práctica de las ideas creativas en la historia de la poesía moderna. El teórico de la literatura digital N. Wardrip-Fruin dice en su texto *Learning to Read Digital Literature* que la crítica literaria tiene que empezar a hacerse consciente de su lugar: “For our health as a field we need to begin to make appropriate connection with computer science in a less *ad-hoc*[1] way, simply

[1] Wikipedia explica: En ingeniería de software, el término *ad hoc* se utiliza para referirse a la manera de trabajo en donde se busca únicamente lograr un desarrollo que dé respuesta al problema en el que se está trabajando, sin dotar al desarrollo de la necesaria modularidad que permita reutilizar sus componentes en el futuro.

recognizing that our discipline is closely connected to C[omputer] S[cience]." [Wardrip-Fruin, 2010, p. 254].

En el mismo sentido va lo escrito por Espen J. Aarseth:

> The main problem seems to be the assumption that cybernetic sign processes can be understood and classified by observing their surface expressions alone. When the relationship between surface sign and user is all that matters, the unique dual materiality of the cybernetic sign process is disregarded. Without an understanding of this duality, however, analyses of communicative phenomena involving cybernetic sign production become superficial and incomplete. [Aarseth, 1999, pp. 39-40]

Es exactamente en este punto que las investigaciones importantes, como el trabajo de Talan Memmot en *Digital Rhetoric and Poetics: Signifying Strategies in Electronic Literature* (2011), o el "Análisis retórico" de Cheri Crenshaw en *Exploiting Kairos in Electronic Literature* (2008), están incompletas. Según lo mencionado por Aarseth, uno debe ser cuidadoso acerca de la aplicación de los conceptos como "interactividad" e "hipertextualidad", porque la mayor parte de estos términos son construcciones ideológicas y operan textualmente en lugar de analíticamente, ya que connotan varias ideas vagas sobre las pantallas de la computadora, la libertad del usuario y los medios de comunicación personalizados, mientras que denotan "nada" [Di Rosario, 2011, p. 89]. Por otra parte, los límites borrosos entre lector y escritor, donde "the medium lends itself to experimental practice, especially to forms that disrupt traditional notions of stable subjectivities and ego-centred discourses"

[Hayles, 2008, p. 17] invitan a hacer un estudio comparativo en profundidad sobre la creatividad egocéntrica y la creatividad colectiva-interactiva. Desde Poincaré -a principios del siglo XX- hasta Whitehead, para quien “la creatividad” aparece como la categoría fundamental de su esquema metafísico que construye puentes entre la literatura, la ciencia y las artes, hasta ahora, el término “creatividad” se ha utilizado, interpretado y reinterpretado ampliamente. Cualquier estudio sobre la política y la economía de un nuevo movimiento artístico implica una investigación sobre las condiciones y las funciones de la creatividad dentro de dicho movimiento.

Los nuevos medios y la poesía

Nuevos teóricos de los medios, como Lev Manovich, declaran que el lenguaje de los nuevos medios está compuesto, en gran parte, de elementos de otras formas culturales ya familiares [Manovich, 2001, p. 71]. Este hecho nos da la posibilidad de un estudio comparativo, en este caso, entre un fenómeno contemporáneo llamado poesía digital (o E-poesía o algunas otras variaciones del término “poesía” en los medios electrónicos) y el discurso histórico de la poesía (sobre todo dentro de los medios impresos). Tal investigación, según lo descrito por Aarseth, debe evitar las retóricas comerciales comunes (como la no-linealidad o la interactividad) en la descripción de la poesía digital, a pesar de que han sido ampliamente aceptadas por los académicos de manera acrítica [Di Rosario, 2011, p. 90].

La poesía es tal vez el término más mistificado en los estudios literarios. Por un lado, se dice que la poesía se escapa de cualquier definición y, por otro, ha habido enormes esfuerzos para formular una definición. La aplicación de este término a un nuevo género literario en medios digitales se ve sometida a esta inseguridad e inexactitud. Los críticos literarios, los filósofos y los poetas aplican diferentes perspectivas que incluso llegan a ser opuestas en la comprensión y la definición de la poesía. Carlos Drummond de Andrade, mediante su poema “Búsqueda de la poesía”, el cual pretende ser una definición, trata de dar a entender lo que es la poesía de una manera negativa. Afirma que, para la poesía, las atracciones, aniversarios, los incidentes personales, la perfección del cuerpo, lo que se piensa o se siente, la naturaleza misma o el hombre en sociedad no importan. Uno debe evitar dramatizar o invocar, investigar y mentir. El poema es el que llama al poeta, es decir, un poema está esperando su turno para ser escrito y mientras vive solo y mudo en “estado de diccionario”[2]. El poeta debe ser paciente y no adular al poema. Aceptando esta situación, el poema encontrará su propia forma, una forma final y concentrada en el espacio. Cuando un poeta mira las palabras del poema, verá en cada una de ellas mil caras secretas bajo el rostro neutral de una pregunta sencilla: ¿has traído la llave?

María Zambrano escribe: “La poesía no puede establecerse a sí misma, no puede definirse a sí misma. No puede, en suma, pretender encontrarse, porque entonces se pierde.” [Zambrano, 2012, p. 121]

[2] http://www.materialdelectura.unam.mx/index.php/poesia-moderna/16-poesia-moderna-cat/107-045-carlos-drummond-de-andrade?start=13

Este tipo de enfoques recuerdan las discusiones sobre el cine donde se trata de definir la experiencia cinematográfica de manera independiente, es decir, sin arraigarla en el teatro, en la pintura, en la literatura o incluso en la fotografía. Bresson, en busca de una verdad cinematográfica, afirma que ésta “no puede ser la verdad del teatro del siglo XX, ni la verdad de la novela, ni la verdad de la pintura”. [Bresson, 1977, p. 5]

Los críticos como André Bazin se concentran en la cuestión de la realidad cinematográfica y la representación de la realidad mediante la edición. Para Bazin, la imagen en el cine es evaluada no según lo que añade a la realidad, sino lo que revela de ella. Este autor también enfatiza que, para muchos creadores como Sergei Eisenstein, la edición es el aspecto que define a una película. Tal perspectiva coloca la experiencia del cine en la distancia que se crea entre la realidad que se despliega ante los ojos de la cámara y el producto en la pantalla. Existen tres principales elementos tecnológicos en esta distancia: la cámara, los dispositivos y la pantalla de edición. Por lo tanto, si el cine va a defenderse a sí mismo como un arte, debe trabajar sobre las diferencias de entrada y salida de cada dispositivo. En el cine, el dispositivo no tiene un sentido propio y el artista como agente de la experiencia artística o como consciencia del dispositivo tiene casi un control totalitario sobre el manejo de la cámara, los dispositivos y la pantalla de edición.

Esta lucha revela la vieja distinción filosófica entre ciencia y tecnología o la profundidad ontológica de la técnica. Aunque esta distinción ha ido desapareciendo después de Nietzsche, aún podemos concentrarnos en las fronteras de difuminación

de la esencia y la apariencia o de la ciencia y la tecnología para descubrir puntos engañosos causados por estas dualidades.

La creatividad digital: una aproximación

En su libro *Prehistoric Digital Poetry*, C. T. Funkhouser considera a la poesía digital como un nuevo género de arte literario, visual y sonoro, que los poetas comenzaron a practicar por medio de las computadoras a finales de la década de 1950. En su análisis, demuestra que los fundamentos de la poesía digital, mecánica y conceptualmente fueron construidos en las décadas previas al surgimiento de las computadoras personales, y se establecieron con firmeza en la década de 1990, antes de que la WWW entrara a formar parte del dominio público. Funkhouser afirma que el estado actual de la poesía digital no puede ser entendido cabalmente sin considerar sus orígenes:

> Digital poetry is an evolving process, employing various techniques that began to form well before the advent of the personal computer and continues to refine itself in today's WWW environment. Poets continue to explore a variety of computerized techniques, from interactive installations to randomized and visual attributes. [Funkhouser, 2007, pp. 1-2]

En su descripción de los orígenes de la poesía digital, Funkhouser observa que los poetas inicialmente utilizaban programas para generar una base de datos y una serie de instrucciones para establecer los contenidos y las formas de una obra. Fue después de mediados de la década de 1960, cuando surgieron los componentes gráficos y cinéticos. Entonces, las computadoras fueron capaces de representar lenguajes

gráficos como poemas en las pantallas o en los medios impresos. Desde esas fechas, se han producido videopoemas y otros tipos de poemas cinéticos que utilizan técnicas y herramientas digitales. A principios de la década de 1980, el hipertexto comenzó a desarrollarse en sintonía con la creciente disponibilidad de computadoras personales. Algunas otras formas experimentales, como la poesía sonora, aparecieron paralelamente, con nuevos avances técnicos. Funkhouser describe:

> When the WWW emerged, multimedia, transcontinental, hyperlinking poets began to spark expression through interconnected motherboards; the status of the art form has risen with the increasing affordability of computing and capabilities of network technologies. Only a few works of digital poetry titles are now circulated offline (few people are publishing digital poetry on diskette or even CDROM). The copious amount of material delivered to readers through the WWW is strong evidence that computers and telecommunications networks heighten the audibility and visibility of this strand of contemporary poetry. [Funkhouser, 2007, p. 2]

Aunque es obvio que un poeta contemporáneo tiene tecnologías más avanzadas a su disposición, Funkhouser demuestra que muchas de las técnicas literarias utilizadas en poemas disponibles en la WWW no pueden clasificarse como novedades literarias. Reiteramos: “the digital techniques used to present them were cultivated in the decades prior to the WWW”. [Funkhouser, 2007, p. 3]

Los factores que Funkhouser utiliza para describir el estado del arte en la poesía digital corresponden a los agentes que Mihaly Csikszentmilahyi reconoce en su descripción del proceso de creatividad y de la producción de obras creativas. En el caso de Funkhouse podemos identificar: 1) cultura (en nuestro caso particular, la cultura digital y el desarrollo de las máquinas que la sustentan), 2) campo o dominio

(las instituciones y las formaciones del poder en este género) y 3) el individuo (el poeta digital).

En su modelo, Csikszentmilahyi afirma que la creatividad resulta de la operación dinámica de un sistema compuesto de tres elementos:

> a culture that contains symbolic rules, a person who brings novelty into the domain, and a field of experts who recognize and validate the innovation. [Csikszentmihalyi, 1999, p. 6].

Csikszentmihalyi explica que las interconexiones que se muestran en su modelo de sistemas se basan en "enlaces dinámicos de causalidad circular" [Csikszentmihalyi, 1999, p. 51]. Por lo tanto, "el punto de partida de este mapa es puramente arbitrario". En fin, "cada uno de los tres sistemas principales: persona, campo y dominio – afecta a los demás y es afectada por ellos a su vez" [Csikszentmihalyi, 1999, p. 51]. Cada componente es un factor necesario en la creatividad, pero no es suficiente en sí mismo para producir novedades.

Csikszentmihalyi sostiene que, para comprender la creatividad totalmente, tenemos que abandonar la visión ptolemaica de la creatividad, en la cual la persona está en el centro de todo, y entonces adoptar un modelo más copernicano, en el que la persona es parte de un sistema de influencias e informaciones mutuas [Csikszentmihalyi, 1999, p. 336].

El dominio, en términos del modelo de Csikszentmihalyi, es el sistema de símbolos que la persona y otras personas que trabajan en el área utilizan. El dominio está compuesto por las convenciones, el conocimiento, el sistema de códigos simbólicos y las técnicas en que la persona debe sumergirse, a fin de que nuevas

variaciones puedan realizarse. El dominio también incluye a todos los productos creados que han sido aceptados por el campo en el pasado. [Sawyer, 2006, p. 216]

Csikszentmihalyi también sugiere algunas maneras importantes en que el dominio puede contribuir al sistema creativo. Entre éstas se encuentra la claridad de la estructura, es decir, qué tan bien organizada está la estructura como sistema de símbolos y qué tan central es dentro de la cultura. En otras palabras, su lugar dentro de las jerarquías culturales con las cuales debe competir para su financiamiento y su accesibilidad. ¿Qué tan fácilmente se puede transmitir de una persona o un productor cultural a la siguiente persona o productor cultural?

El campo, por el contrario, es la organización social, la jerarquía de grupos y personas que tratan y pueden influir en el sistema de conocimientos, el dominio cultural específico, sobre una base regular. El campo es una red compleja de expertos, con diferentes conocimientos, estatus y poder [Sawyer, 2006, p. 216].

Csikszentmihalyi sostiene, además, que el campo contribuye al sistema creativo eligiendo un filtro amplio o estrecho para seleccionar novedades. Es decir, puede permitir un gran número y variedad de nuevos cambios en el dominio o restringir las perspectivas de éxito en la producción de obras originales con sólo seleccionar un número limitado de cambios en el dominio. El campo también puede ser conservador o aventurero, ser reactivo o proactivo ante novedades. O puede hacer ambas cosas en momentos diferentes, dependiendo de las circunstancias que se den dentro de la sociedad en general.

Si vamos más allá de la visión de la creatividad centrada en la persona, tenemos que realizar un análisis de la creatividad en la poesía digital ligada a su dominio y a su cultura. Por lo tanto, una lectura de la historia de los medios digitales parece necesaria. Es decir, un repaso del desarrollo de las computadoras y de los lenguajes de programación, sus políticas y sus terminologías. Para tener una referencia comparativa fuera del campo, también es necesario considerar los juegos de computadora como un referente necesario para analizar el desarrollo de la poesía digital.

El poeta hacker: la creatividad juguetona

Adalaide Morris trae a colación una pregunta exacta cuando escribe: “are the terms Aristotle developed to describe drama useful, as Brenda Laurel and others claim, in the study of human-computer activity? Is it productive, as Janet Murray and others hold, to subsume the study of computer games into the discipline of narratology?” [Morris, 2006, p. 5] Ciertamente, en cualquier aproximación a la poesía digital, el “análisis específico de medios de comunicación” de N. Katherine Hayles o el “materialismo digital” de Lev Manovich nos llevarán a diferentes nociones de poética. Morris hace hincapié:

> Of the three inseparable components of a digital poem—data fields, code, and display—only the final element, the display, is immediately visible. For this reason, critics of print poetry often underestimate the otherness of new media writing, its resistance, excess, or supplementary to print poetics, even to experimental poetics. [Morris, 2006, p. 9]

La historia de los juegos de computadora es, en parte, una historia de la tecnología que utilizan. El juego de computadora requiere tecnología capaz de manejar grandes cantidades de datos y de representaciones de estos datos. La relación entre un fenómeno tecnológico, como las computadoras, y la cultura no es una relación simple: algunas teorías afirman que la tecnología determina la cultura, otras afirman que la cultura determina la tecnología. Puede ser más razonable ver este asunto como una historia de influencias mutuas, donde la tecnología puede inspirar (o habilitar) desarrollos culturales y los desarrollos culturales pueden inspirar nuevas tecnologías. Aarseth señala:

> the cybertext reader is a player, a gambler; the cybertext is a game-world or world-game; it is possible to explore, get lost, and discover secret paths in these texts, not metaphorically, but through the topological structures of the textual machinery. [Aarseth, 1997, p. 4]

La poesía, en su forma tradicional, no puede nunca tomar la forma de un videojuego porque los videojuegos, como los conocemos, en su forma popular (es decir, un cúmulo de acciones rápidas, a las cuales el jugador responde físicamente), son la antítesis de los propósitos de un determinado estilo de poema muy frecuente en lapoesía tradicional.

A propósito de esta distancia, Steven J. Brams escribe:

> Situated, as they are, in different worlds, game theory and literature have their own coordination problem, with game theorists and humanists not often benefiting from each others' insights. What makes a literary creation succeed is not just its overall structure but also its details, including the emotional lives of its characters. Game theorists need to ponder these and adapt their theory accordingly, just as literary scholars need to appreciate that game

> theory has its own richness that goes beyond mathematical symbols and abstract forms. [Brams, 2011, p. 27]

Es decir, a pesar de que los videojuegos y la literatura pertenecen a mundos diferentes, existe la posibilidad de que haya una coordinación fructífera entre ellos, siempre y cuando haya disposición de las partes para aprender las peculiaridades del campo que desconocen.

En las *Investigaciones filosóficas*, Ludwing Wittgenstein afirma que "el problema de la naturaleza del arte es como la naturaleza de los juegos" [Wittgenstein, 1972, p. 31]. Un juego, según el filósofo francés Roger Caillois, debe tener las siguientes características:

> 1. Debe ser lúdico.
> 2. Es limitado en tiempo y espacio.
> 3. Su resultado no es claro, es incierto.
> 4. No es productivo. Esto significa que su resultado no puede ser algo útil para ser vendido.
> 5. Debe tener reglas o gramática.
> 6. Debe contener otra realidad. Una realidad ficticia que es diferente de la realidad cotidiana. [Caillois, 2006, p. 128]

A lo anterior pueden sumarse las características básicas que atribuye el diseñador de juegos Chris Crawford a los juegos de computadora:

> 1. Representación: un juego es un sistema formal y cerrado que subjetivamente representa un subconjunto de la realidad.
> 2. Interacción: el juego reconoce y reacciona ante el jugador.
> 3. Conflicto: un juego presupone un conflicto.
> 4. Seguridad: el jugador está seguro (en un sentido literal) de los acontecimientos en el juego. [Crawford, 1997, pp. 3-9]

La Teoría de Juegos estudia de manera formal y abstracta las decisiones que deben tomar diversos adversarios en conflicto para optimizar sus resultados. Esta disciplina puede definirse como el estudio de modelos matemáticos que describen el conflicto y la cooperación entre entes inteligentes que toman decisiones. Tales decisiones se consideran estratégicas. Es decir, los entes que participan en el juego actúan teniendo en cuenta las acciones que tomarían los demás. Según Lidia Morales Benito en *Las reglas del juego: teoría y práctica del juego en literatura*:

> Buen ejemplo de instrucciones previas a la actividad lectora es el prefacio de *Rayuela* de Julio Cortázar, quien propone dos posibles procedimientos, el lineal y el lúdico: El primero se deja leer en la forma corriente, y termina en el capítulo 56, al pie del cual hay tres vistosas estrellitas que equivalen a la palabra "Fin". Por consiguiente, el lector prescindirá sin remordimientos de lo que sigue. El segundo se deja leer empezando por el capítulo 73 y siguiendo luego en el orden que se indica al pie de cada capítulo.
>
> Antes de iniciar la lectura de Rayuela, el jugador deberá aceptar una de las dos propuestas y, si desea lanzarse a una aventura lúdica, se verá obligado a dejarse conducir por la secuencia de capítulos preestablecida por el autor. […] Al igual que el autor-jugador, el lector que participa en este tipo de actividades lúdicas también podría ser calificado de "aguafiestas" o de "tramposo" según su actitud frente a las reglas impuestas por el escritor. [Morales Benito, 2012, pp. 272-276]

Las obras de teóricos del juego como Steven J. Brams y las investigaciones relativas a la semántica de la teoría de juegos nos proporcionan las herramientas analíticas adecuadas para la descripción de técnicas literarias (técnicas relativas a la trama, la ambientación, la narrativa, el estilo, el tema y la subjetividad). En un terreno tan abstracto e incluso pragmático, estos estudios pueden conducir a llevar a cabo una investigación comparativa sobre la creatividad en la poesía digital y en la poesía

impresa, especialmente cuando se trata de considerar la función del lector en los nuevos medios. Dice Brams:

> The game played between the author and the reader, as the reader progressively acquires more information (not necessarily accurate, such as the false clues in a mystery), is one that does not seem to have been analyzed for any literary work. An appropriate framework for such an analysis might be the theory of psychological games (Geanakoplos, Pearce, and Stacchetti 1989) or "information-dependent games" (Gilboa and Schmeidler 1988), in which the players' payoffs depend on whether certain postulated beliefs are fulfilled. [Brams, 2011, p. 26]

O sea, mientras la cultura impresa considera funcionalidades separadas para el lector y el escritor, y por lo tanto, como enfatiza Brams, parece no haber analizado varios juegos existentes entre el autor y el lector en ninguna obra literaria, los nuevos medios se centran en la naturaleza lúdica de la interacción entre el lector y el escritor e incluso visualizan o encarnan tales discursos interactivos lúdicos.

La hegemonía de la imprenta y de las técnicas de impresión todavía sigue dominando a las academias, de tal manera que parece natural investigar sobre las técnicas de impresión que hacen que una obra en soporte de papel sea posible, pero no así investigar y analizar los códigos y estructuras que permiten construir los poemas digitales. Sin embargo, para las generaciones que han crecido en un mundo con desarrollo cibernético, el lenguaje natural y el lenguaje de programación conviven con relativa comodidad. Como ejemplo podemos mencionar la poesía-código que se escribe utilizando los comandos de lenguaje de programación y su doble sentido dentro del lenguaje natural. Veamos el siguiente poema-código de Jason Kopylec escrito con la sintaxis de Java:

TWOFACED

```
public class TwoFaced {
        public String greet() {
                return “Hi! So great to see you!”;
        }
        private String think() {
                return “Fucking bitch.”;
        }
}
```

Este texto representa un objeto digital que puede referirse a una persona o una máquina. El objeto se nombra como un ser de dos caras. Tiene una funcionalidad pública y otra privada. Saluda públicamente a los demás diciendo "¡Hola! ¡Qué bueno verte!" y en privado piensa para sí mismo "Put@ de mierda". Una vez que el programa entra en acción, la funcionalidad privada solo es visible para el objeto digital y nadie más. Cualquier persona fuera de este objeto solo puede ver el saludo público. Por tanto, este texto puede interpretarse como una representación de la hipocresía en la comunicación humana.

Observar la superficie de un poema o de una narrativa digital y analizar los códigos fuente de las obras digitales son tareas que pertenecen a diferentes discursos. Los códigos fuente de cada obra tienen sus propias retóricas que revelan la subjetividad del programador, su política específica para utilizar un determinado lenguaje de programación en lugar de otro, sus preocupaciones, sus experiencias y otros

elementos creativos totalmente distintos de lo que la misma obra muestra en la superficie. Los códigos fuente son considerados como el cuerpo de la poesía digital.

El corpus del estudio

Un poema digital puede tomar varias formas: hipertexto, poesía cinética, animación generada por computadora, poesía visual, poesía interactiva, poesía codificada, poesía holográfica, poesía experimental en video, poesía generativa, poesía sonora, poesía colaborativa en red o instalación, etc. Puede incluir cualquier mezcla o híbrido de imagen, texto, sonido, código, espacio o incluso parte del cuerpo humano. La gama de posibilidades que las combinaciones de estos elementos producen es amplia y este estudio no puede abarcarlas todas. Sin embargo, muchos de los aspectos fenomenológicos importantes de la retórica digital son compartidos por estas diferentes posibilidades: procesualidad (incompletitud, trabajo abierto); interactividad, hipermedialidad (integración, convergencia) y situación en redes (interacción, colaboración) [Torres, p. 5]. Por lo tanto, este estudio se centrará principalmente en tipos textuales como la poesía generativa, Twitter e hipertextual. Esto proporcionará las bases para situarse en el límite entre la poesía impresa y la poesía digital y observar la poesía digital desde dos puntos de vista diferentes: el digital y el impreso.

En este estudio se consideran tres categorías principales: la poesía generativa (como la forma más antigua de poesía digital), el hipertexto (una de las formas más

populares del género) y la poesía de Twitter (que se presenta como una poesía generativa, hipertextual y en red).

Los códigos fuente de las obras provienen de tres antologías publicadas por la *Electronic Literature Organization* (ELO). También he agregado algunos ejemplos de editores independientes, como sucede con el programa *Storyspace*, que fue uno de los pioneros en el campo de la poesía hipertextual. Asimismo abordaré los desarrollos recientes en lingüística computacional, que involucran redes neuronales, pues trajeron algunas posibilidades interesantes al campo de la poesía generativa. Algunas de las obras han sido analizadas en varias dimensiones y otras sólo se mencionan como puntos comparativos o como ejemplos de un uso retórico o algorítmico.

Así, el corpus de estudio está integrador las siguientes obras, clasificadas de acuerdo conlascategorías mencionadas:

Poesía Generativa

Obra	Autor	Lenguaje de programación	Editor
Argot Ogre, OK!	Andrew Plotkin	JavaScript	ELO3
Contemporary Japanese Poetry Generator	SHINONOME Nodoka	JavaScript	ELO3
Frequency	Scott Rettberg	Ruby y JavaScript	ELO3
PoetRNN	Sam Ballas	Python	Independiente
Poems: RNN	Denis Krivitski	Python	Independiente

Poesía en Twitter

Obra	Autor	Lenguaje de programación	Editor
Everyword	Allison Parrish	Python	ELO3
poem.exe	Liam Cooke	Ruby	ELO3
ROM_TXT	Zach Whalen	Python	ELO3

Poesía Hipertextual

Obra	Autor	Lenguaje de programación	Editor
High Muck a Muck	Fred Wah y otros	HTML5	ELO3
Stir Frys	Jim Andrews	JavaScript , HTML/CSS	ELO1
Self Portraits	Talan Memmott	JavaScript, HTML/CSS	ELO1
Lexia to Perplexia	Talan Memmott	JavaScript, HTML/CSS	ELO1
Oulipoems	Millie Niss with Martha Deed	JavaScript, HTML/CSS	ELO1
The Last Performance	Judd Morrissey	PHP, HTML/CSS	ELO2
Patchwork Girl	Shelley Jackson	Storyspace	Easygate

Para evitar una innecesaria complejidad, seleccioné las obras que comparten lenguajes de programación similares. Por ejemplo, evité intencionalmente todas las obras hipertextuales escritas en Flash o Director. Menciono *Storyspace* como una referencia histórica y también tomo en consideración las tendencias existentes que apuntan al futuro de la poesía digital. Java, Python y Ruby son ahora los lenguajes de programación más comunes entre los poetas digitales para las tres categorías de este

estudio y también para la industria que actúa en este ramo. C, Basic, Pascal y otros lenguajes avanzados ya no se utilizan como lenguajes de propósito general.

La estructura de la tesis

En el campo de la poesía digital, los enfoques que se originan a partir de la poesía impresa parecen problemáticos. En la poesía digital, nos enfrentamos a un nuevo tipo de textualidad, que implica nuevas estrategias de lectura. Di Rosario cita a Aerseth cuando explica que en la poesía digital el texto funciona como una máquina material, un dispositivo capaz de manipularse a sí mismo y también al lector. En la literatura impresa, el texto no es capaz de manipularse a sí mismo. Por lo tanto, el primer capítulo de esta tesis trata de destacar las principales características de este tipo de texto y las teorías útiles para trabajar con esa peculiar textualidad. Además, en estas circunstancias la materialidad de un texto es parte del fenómeno del texto. Las computadoras, hardware y software, forman la materialidad de cualquier texto en la literatura digital. Este capítulo explica la interconectividad entre la retórica digital, los medios digitales y el poema digital en su conjunto.

El texto electrónico como un todo se analiza a nivel ontológico en el segundo capítulo. Una ontología es una especificación explícita de una conceptualización. Para la poesía digital, dicha ontología debe ser válida tanto en el nivel fenomenológico, donde el texto se considera un poema, como en el nivel computacional, donde el texto es un documento digital. La totalidad del fenómeno de

la poesía digital reside en esta existencia híbrida. La potencia de un poema digital está relacionada tanto con las dimensiones de la novedad y la creatividad que aporta a la experiencia textual como con su materialidad, su existencia digital, que permiten la no linealidad, la interactividad, etc. Por lo anterior, en el estudio, utilizo un híbrido de enfoques que refleja dicha ontología: cuando se trata del autor, del individuo y su creatividad, uso el enfoque de C.S. Peirce, y cuando se trata de creatividad social, colaborativa o no centrada en una persona, aplico el enfoque de Csikszentmihalyi. Además, en el mundo digital todo es información y la existencia se manifiesta como información constituida por números digitales. Es decir, si una persona no está registrada en el sistema de una universidad o una organización, no existe para dicha universidad u organización. La realidad digital también manipula la realidad real para las personas que la interpretan como tal. Por ejemplo, *Pokemon Go*, un juego de computadora muy popular, asume que hay insectos en el espacio físico y que la gente puede seguirlos y matarlos. Los usuarios utilizan la cámara de su móvil que representa la realidad real y el juego muestra dónde se distribuyen los insectos digitales en el espacio. En este juego, esos insectos son ontológicamente inexistentes para cualquiera que no interprete la realidad digital como una realidad verdadera. Una realidad aumentada —una experiencia interactiva de un entorno del mundo real en el que los objetos que residen en el mundo real son "aumentados" por información perceptiva generada por computadora— tiene un orden ontológico híbrido. No se las puede llamar simplemente una ilusión. Por lo tanto, en el capítulo dos de la tesis, mi

ontología híbrida proporciona un enfoque sistemático hacia el poema digital que es consistente tanto en la poética digital como en la tradicional.

En el tercer capítulo de la tesis se aplican las teorías y las ontologías de los capítulos anteriores al corpus. Ahí sitúo en primer plano el impacto de los algoritmos, la estructura del código fuente y la materialidad de un poema digital en el desarrollo de la creatividad en el poema.

I

Las configuraciones del texto electrónico

Como mencioné antes en la introducción, los textos digitales son textos dinámicos. Es decir, en contraste con los textos estáticos que no cambian con el tiempo, pueden desaparecer o volverse disfuncionales. Son efímeros. Este tipo de textos requiere una nueva semiótica propia. En cualquier texto digital, coexisten tres entidades: el programador o el autor, la máquina y el lector. Cada una de estas entidades trabaja con una semiótica diferente a la otra. Un programador trabaja con lenguaje informático, legible por humanos; una máquina, con lenguaje de máquina no legible por humanos; y un lector, con lenguaje natural que se acerca al texto digital. La mayoría de los estudios sobre textos electrónicos se han enfocado principalmente en la experiencia del lector. En cambio, en este estudio se trata de abordar el texto en su totalidad, lo cual implica tres categorías de semiótica.

En semiótica, se analiza el significado utilizando una abstracción llamada signo. Según C. S. Peirce, un signo, siguiendo la definición clásica de los estoicistas, es algo que está en lugar de otra cosa, que no necesariamente existe, y se dirige a alguien. Siguiendo al teórico norteamericano, cualquier signo consta de tres elementos: un representante, la parte material del signo; un objeto, el referente al que se refiere el signo; y un interpretante, que se deriva o genera por el signo. También es importante notar que el interpretante en sí es otro signo. Es decir, el significado se construye mediante un valor, pero mediante una función recursiva, es decir, un

proceso continuo de interpretación. El proceso se detiene temporalmente, cuando el intérprete no siente la necesidad de continuar la serie de interpretaciones.

De acuerdo con ese orden de ideas, en cualquier texto digital dos extremos del texto son humanos y entre ellos reside la máquina. Para la semiótica, un símbolo tiene una relación arbitraria con el objeto. El significado se puede producir al traer elementos significativos que tienen relaciones impredecibles con el signo. Además, el significado humano puede detenerse en cualquier momento que lo desee. Sin embargo, la máquina no puede manejar relaciones impredecibles con el signo y cualquier proceso de detención o pausa debe definirse de antemano con precisión.

Para entender mejor lo arriba mencionado, tomo prestada la idea de *techno-texts* o máquina textual de Katherine Hayles, donde se sugiere que para comprender los textos digitales es necesario un análisis específico de los medios. Giovanna Di Rosario explica a Hayles de la siguiente manera:

> Katherine Hayles insists on the necessity of studying the specific materiality of the support or better she suggests the MSA – Media Specific Analysis. Hayles argues that a text's instantiation in a particular medium shapes it in ways that cannot be divorced from the meaning of its "words (and other semiotic components)" and calls for the need to develop a theory that takes into consideration the medium as a crucial aspect of the content of a work. Hayles writes: "the physical attributes constituting any artefact are potentially infinite […]. From this infinite array a technotext will select a few to foreground and work into its thematic concerns. Materiality thus emerges from interactions between physical proprieties and a work's artistic strategies. For this reason, materiality cannot be specified in advance, as if it pre-existed the specificity of the work. [Di Rosario, 2012, p. 8]

Philip Brey, en su ensayo epistomológico sobre la interacción humano-computadora, ofrece una terminología que es muy fructífera para nuestro estudio. Los expertos

consideran que la computadora es un tipo de *artefacto cognitivo* capaz de extender una amplia gama de capacidades cognitivas de los seres humanos. Su sugerencia está arraigada en la teoría de McLuhan, que considera a los medios como una extensión de lo humano. A propósito de lo anterior, Brey menciona a Donald Norman, un psicólogo que acuñó este término. Según Norman, un artefacto cognitivo tiene la capacidad de representar, almacenar, recuperar y manipular información. Estos artefactos están diseñados para mantener, mostrar u operar información que sirve a una función representativa. Brey clasifica cuatro funciones de los artefactos cognitivos: memoria, interpretación, búsqueda y pensamientos conceptuales [Brey, pp. 384-385] y explica:

> 1. MEMORY
> Human memory is the psychological faculty by which we store information and retrieve it for later use. Cognitive artifacts that extend memory functions may be called memory devices. They are artifacts that help us encode, store and retrieve information.
>
> 2. INTERPRETATION
>
> Interpretation is the ability to assign meanings to input data, through the assignment of one or more concepts or categories. Interpretation can be qualitative or quantitative. Quantitative interpretation is the assignment of a numerical value to a perceived quality. Another word for this is measurement. Qualitative interpretation is the assignment to data of a qualitative concept or category. There are many cognitive artifacts that aid in the qualitative interpretation by giving criteria, templates or examples for the application of a concept.
>
> 3. SEARCH
>
> When we interact with the world, we often actively look for things that we are trying to locate but have not observed yet. We constantly look around for people, pens, purses, stores, food, stamps, road signs, words, barcodes, and numerous other things that we need to see, locate or use. The ability to search

> and subsequently recognize things is one of our fundamental cognitive abilities.
>
> 4. CONCEPTUAL THOUGHT
>
> Conceptual thought is the ability to arrive at new conceptual structures (ideas or beliefs) through the modification (analysis or synthesis) of existing ones. Conceptual thought often involves problem solving: it often involves cognitive goals like finding the solution to a mathematical equation, determining the best way to furnish a room, finding an adequate translation into English for a sentence in Spanish, or thinking up the most diplomatic answer to a potentially embarrassing question. [Brey, p. 385]

Todas estas cuatro funciones son las funciones de las computadoras modernas. Esta clasificación funcional también nos proporciona una categorización de los algoritmos utilizados en ciencias de la computación. En poesía generativa, poesía de Twitter o poesía de hipertexto, estos son los elementos básicos que diferencian un poema de otro, en el nivel de algoritmo: ¿Qué algoritmo de búsqueda se está utilizando? ¿Cómo el programa restaura y recupera los datos? Y así sucesivamente.

Teniendo en cuenta esta discusión, no es sorprendente que hablemos ahora de alfabetización digital. De la capacidad de manejar tal artefacto cognitivo que para los usuarios y analistas debe definirse de manera diferente: un usuario debe poder usar el artefacto para sus fines, pero un analista debe poder explicar cómo funciona el artefacto. Debe tener un conocimiento del algoritmo, de la codificación, y del software y el hardware.

En el primer volumen de la antología de literatura electrónica publicada por ELO, Jim Andrews, autor de *Stir Fry Texts*, escribe: “I did the programming in DHTML. I am indebted to Marko Niemi for his upgrading of the programming in

2004. Now they run OK on both PC and Mac and most contemporary browsers on both platforms." En el mismo volumen, Talan Memmott, en la descripción de su obra *Lexia to Perplexia*, afirma: "The work makes wide use of DHTML and JavaScript." Y Giselle Beiguelman explica su *Code Movie* usando esta expresión: "Code Movies are made with hex, ASCII, and binary codes extracted from JPG images. Saved as simple text, they are reworked and edited in Flash."

En el segundo volumen de la misma antología, otro autor, Judd Morrissey, añade la siguiente expresión a su obra *The Last Performance*: "For Windows users, Firefox is recommended. Internet Explorer versions 6 and below are not supported." Además, la antología también publica cada obra con un comentario como este: "Creative Commons Attribution-NonCommercial-NoDerivs 2.5 License." En el campo tradicional de la literatura impresa, tales afirmaciones y expresiones no tienen ningún significado específico. Sin embargo, al hablar de la literatura electrónica en general y de la poesía digital, en particular, no hay escapatoria de estas terminologías. Cualquier estudio de tal género de la poesía implica el uso de estas palabras.

Por todo lo anterior, en este primer capítulo se proporciona una descripción general y una historia del desarrollo de estas características, utilizando el modelo sugerido por Mihaly Csikszentmilahyi descrito en la introducción. Es decir, primero se aborda el tema de la cultura de los medios electrónicos y el dominio del poema digital y en los capítulos siguientes la atención se concentrará en el desarrollo de un poema de un artista específico o de un equipo de artistas. Así, las tres entidades —cultura, campo o dominio e individuo— participan en la configuración y la

reconfiguración de la creatividad en la poesía digital. Para las explicaciones de cultura y dominio utilizo las que ofrece Csikszentmilahyi, las cuales ya he esbozado, y para explicar la creatividad en una obra de poesía digital específica, utilizo los enfoques cercanos pero diferentes que Whitehead y Peirce aplicaron en sus filosofías. Peirce mira la creatividad como una continuidad mientras Whitehead lo hace a través de su punto de vista atómico, es decir, la aborda como discontinuidad. El segundo capítulo proporciona un híbrido de estos enfoques para explicar cómo funciona la creatividad en un poema digital en un nivel ontológico.

I-I

Primera configuración: la cultura digital

La enciclopedia libre, Wikipedia, un producto de los nuevos medios, define los nuevos medios de comunicación como formas que son nativas de las computadoras y que confían en las computadoras para su distribución. Algunos ejemplos de los nuevos medios son los sitios web, las aplicaciones de teléfonos móviles, juegos digitales, animaciones e instalaciones interactivas y digitales. Los nuevos medios no incluyen los programas de televisión (como emisión analógica), largometrajes, revistas y libros, a menos que contengan tecnologías que permitan procesos digitales generativos o interactivos.

El trabajo de Marshall McLuhan sobre la comprensión de los medios, junto con movimientos como el estructuralismo o la teoría del lenguaje de Saussure e incluso el formalismo ruso, son todos productos de un siglo que responden a una historia que se definía más por los contenidos que por las estructuras. Parece necesario comenzar con McLuhan cuando intentamos analizar un discurso en los estudios de los medios de comunicación. Para McLuhan, un medio de comunicación ya no es una herramienta simple para transmitir un mensaje a un receptor sino una extensión del hombre. Cada medio de comunicación amplifica o acelera los procesos existentes y por lo tanto trae un cambio de escala o ritmo o forma o patrón en la asociación, los asuntos y la acción humana. Los resultados traen algunas

consecuencias psíquicas y sociales. El ejemplo de McLuhan del foco es famoso: la luz eléctrica no tiene ningún contenido; por lo tanto, nadie la había considerado como un medio de comunicación. Pero sucede que tuvo un efecto social y cambió la concepción humana del tiempo y la sociedad. Como McLuhan afirma:

> The electric light escapes attention as a communication medium just because it has no "content." And this makes it an invaluable instance of how people fail to study media at all. For it is not till the electric light is used to spell out some brand name that it is noticed as a medium. Then it is not the light but the "content" (or what is really another medium) that is noticed. The message of the electric light is like the message of electric power in industry, totally radical, pervasive, and decentralized. For electric light and power are separate from their uses, yet they eliminate time and space factors in human association exactly as do radio, telegraph, telephone, and TV, creating involvement in depth [McLuhan, 1994, p. 31].

La definición de McLuhan de los medios relaciona la sociedad (como una estructura) a la cultura (como un contenido). Resumo la clasificación de los puntos principales de la teoría de McLuhan en estas dimensiones:

1. Los medios de comunicación son una extensión psíquica o física del sentido humano; por lo tanto cambian al ser humano extendiendo al hombre.

2. La experiencia de los medios de comunicación y su estructura comunicativa y su configuración afectan a la sociedad y por lo tanto la modifican.

3. Los medios pueden ser categorizados por diferentes grados de participación. Los medios de comunicación fríos son más participativos y los medios calientes son menos. Los medios fríos implican más interacciones

psíquicas y físicas humanas y los medios calientes son menos interactivos y por lo tanto implican menos los sentidos humanos.

4. El contenido de un medio es siempre otro medio. Por lo tanto, la experiencia de la traducción forma parte de la experiencia de cada medio.

5. Vivimos en la era de Marconi, en la era de la electrónica que es la fuerza más revolucionaria de la sociedad. Los grados de participación clarifican los cambios sociales relacionados con el uso de los medios desde la cultura oral hasta la cultura impresa y finalmente a la era electrónica.

En su análisis del juego, McLuhan muestra que este, como cualquier medio de información, es una extensión del individuo o del grupo. Sus efectos en el grupo o el individuo son una reconfiguración de las partes de dicho grupo o individuo que no se han extendido de ese modo. McLuhan explica:

> Any game, like any medium of information, is an extension of the individual or the group. Its effect on the group or individual is a reconfiguring of the parts of the group or individual that are not so extended. A work of art has no existence or function apart from its effects on human observers. And art, like games or popular arts, and like media of communication, has the power to impose its own assumptions by setting the human community into new relationships and postures... Games are extensions, not of our private but of our social selves, and they are media of communication...” [McLuhan, 1964, p. 255]

Tanto como cualquier extensión de lo humano, la cultura digital también define y abarca las formas de pensar y actuar que se encarnan dentro de los medios de

comunicación y la tecnología. La abstracción, la codificación, la autorregulación, la virtualización y la programación ahora dan forma a nuestra cotidianidad.

Hablando de la cultura digital o de los medios digitales utilizo la definición de Charlie Gere de “lo digital”:

> Given how important digital technology has become to our lives it is useful to know what the word “digital” actually means. In technical terms it is used to refer to data in the form of discrete elements. Though it could refer to almost any system, numerical, linguistic or otherwise, used to describe phenomena in discrete terms over the last 60 or so years, the word has become synonymous with the technology which has made much of the aforementioned possible, electronic digital binary computers. [Gere, 2009, p. 15]

En cierta medida, los términos “tecnología informática” y “tecnología digital” se han convertido en palabras intercambiables.

Bolter y Grusin llevaron la teoría de McLuhan a otros puntos sugiriendo los medios de comunicación como remediaciones. Los teóricos explican que la cultura occidental, para representar la inmediatez, intenta siempre esconder los medios de comunicación. En la distancia entre la realidad y la realidad representada, y mediante el uso de las definiciones epistemológicas de las palabras mediación, hipermediación y remediación, tratan de hacer uso de teorías post-estructuralistas, el nuevo marxismo, la teoría postmodernista y la teoría feminista para sugerir una teoría sobre el desarrollo y la interpretación de los nuevos medios. Al desafiar el mito de la inmediatez transparente, definen el concepto de remediación en las siguientes tres categorías que resumo aquí:

1. En primer lugar, la remediación como mediación de la mediación: en la que cada acto de mediación depende de otros actos de mediación. En esta categoría los medios están continuamente comentando y sustituyendo unos a otros.

2. En segundo lugar, la remediación como la inseparabilidad de la mediación y la realidad, en la que los medios de comunicación son reales como artefactos. Incluso la cultura occidental todavía reconoce que todos los medios de comunicación remedian lo real.

3. En tercer lugar, los medios como una reforma, en la que el objetivo de la remediación es remodelar, criticar o reformar otros medios.

Aquí podemos empezar a cuestionar si una extensión de lo humano por medios digitales nos llevará a una condición post-humana o no, condición que explicaré en los siguientes párrafos.

En *Mind Children*, Hans Moravec expresa: “pronto será posible descargar la conciencia humana en la computadora” [Hayles, 2008, p. 1]. Katherine Hayles utiliza esta afirmación para formular tres preguntas centrales en su viaje crítico sobre el concepto de “Post-humano”: La primera versa sobre cómo la información perdió su cuerpo, es decir, la conceptualización de la información como una entidad separada de las formas materiales en las que se suponía que encarnaba. La segunda se refiere a cómo el Cyborg fue creado como un artefacto tecnológico y un ícono cultural

después de la Segunda Guerra Mundial. La tercera, sobre cómo una construcción históricamente específica llamada “ser humano” está dando paso a una construcción diferente llamada post-humana. En la definición del concepto de post-humano, Hayles sugiere cuatro supuestos que resumo de la siguiente manera:

> 1. La visión post-humana privilegia los patrones informativos sobre la ejemplificación material, de modo que la encarnación en un sustrato biológico sea vista como un accidente de la historia en lugar de una inevitabilidad de la vida.
> 2. La visión post-humana considera a la consciencia -que en la tradición occidental fue considerada como la sede de la identidad humana- como un comienzo evolutivo y ve la consciencia como un epifenómeno.
> 3. La visión post-humana piensa en el cuerpo como la prótesis original que todos aprendemos a manipular, de modo que extender o reemplazar el cuerpo con otra prótesis se convierte en una continuación de un proceso que comenzó antes de que naciéramos.
> 4. La visión post-humana configura el ser humano de modo que pueda ser articulado de manera inconsútil con la máquina inteligente.

Si “humano” está alineado con las nociones de iluminación del humanismo liberal y su énfasis está en el “ser natural” y la libertad del individuo, la visión “post-humana” privilegia la información sobre la materialidad, considera la conciencia como un

epifenómeno e imagina el cuerpo como prótesis para la mente. Hayles luego explica: "People become posthuman because they think they are posthuman" [Hayles, 1999, p. 6]. Como hija del matrimonio de la teoría de control y la teoría de la información, la cibernética señaló a tres agentes de gran alcance: la información, el control y la comunicación. Estos factores operan conjuntamente para lograr una síntesis de lo orgánico y lo mecánico, es decir, lo posthumano.

Para Hayles, el sueño de la información es estar libre de las limitaciones materiales que gobiernan el mundo moral. Ella utiliza la teoría de Shannon para describir la información como una función de la probabilidad sin dimensiones, sin materialidad y sin conexión necesaria con el significado: un patrón, no una presencia. En tal contexto la “virtualidad” es una condición, una percepción cultural de que los objetos materiales son interpretados por los patrones de información y, cuando el patrón se vuelve predominante sobre la presencia y nuestra mentalidad cultural acepta tal impresión, entramos en la condición de la virtualidad. Esta condición no es el resultado de la fuerza irresistible de la determinación tecnológica, sino el resultado de negociaciones históricamente determinadas.

Más adelante, Hayles discute el terror de la post-humanidad en términos del remplazo del ser humano. Sin embargo, no acepta las visiones apocalípticas sobre el tema porque el post-humano no necesita ser anti-humano. Para Hayles, el ser humano es ante todo el ser encarnado y las complejidades de esta encarnación significan que la conciencia humana se desarrolla de maneras muy diferentes de las de la inteligencia encarnadas en máquinas cibernéticas. De esta manera, afirma que el

post-humano evoca la perspectiva de salir de algunas de las viejas cajas y abrir nuevas formas de pensar sobre lo que significa ser humano.

I-I-I

La reconfiguración del poema

El informe que proporcioné anteriormente aborda perspectivas de los estudios de los nuevos medios y la cultura digital. En los siguientes capítulos las discutiré con más detalle. Sin embargo, hay que mencionar que el terror apocalíptico que Hayles menciona es una parte del terror que los investigadores clásicos de la cultura impresa enfrentan cuando se encuentran con algunas de las terminologías que voy a discutir en este capítulo. Hay que mencionar que entender los nuevos medios y la poesía digital implica algún conocimiento de este vocabulario. En lo posible, trato de evitar definiciones muy complicadas de cada concepto e intento dar descripciones exactas. El vocabulario común de cualquier historia de los sistemas informáticos necesariamente utilizará terminologías como *sistemas operativos* y *lenguajes de programación*. No se puede utilizar ninguna computadora sin conocerlas. Por lo tanto, antes de analizar algunos aspectos clave de la historia de la informática, explicaré brevemente las dos entidades mencionadas. La historia de los sistemas informáticos se puede leer en detalle en los libros nombrados en mi bibliografía, pero la resumiré brevemente y me concentraré sólo en conceptos que parecen importantes en nuestra discusión sobre la poesía digital.

No es necesario afirmar que un poema digital es un programa de computadora. Es legible o utilizable en una máquina informática. Las enciclopedias como Wikipedia definen un programa de computadora como "una colección estructurada general de secuencias de instrucciones que realizan una tarea específica

cuando son ejecutadas por una computadora. Una computadora requiere programas para funcionar. Un programa de computadora generalmente es escrito por un programador en un lenguaje de programación."

El mismo hecho de que un poema digital sea un programa lo pone en un nivel ontológico diferente al de un poema impreso. Sus materialidades son distintas. Para leer un poema impreso, podemos tomar un libro del estante de una biblioteca. Pero para un poema digital, tenemos que acercarnos a una máquina informática, usar un sistema operativo y luego usar un entorno adecuado, como un navegador, donde el poema, como cualquier otro programa, se puede ejecutar. También tenemos que utilizar dispositivos como el monitor o una pantalla táctil para ver el contenido y un teclado, mouse o los dedos para interactuar con el poema. Leer o interactuar con un poema digital implica una cultura de la lectura diferente y escribirlo también requiere de nuevas aproximaciones, tecnologías y cultura.

Por ejemplo, *Patchwork Girl* es una obra de Shelley Jackson publicada en 1995 y elaborada con un programa llamado Storyspace, producido por Eastgate Systems. Esta empresa de informática, fundada en 1982, antes del surgimiento de la World Wide Web, publica obras literarias digitales para sistemas operativos Macintosh y Windows. Es decir, en las computadoras que utilicen el sistema operativo Linux no se pueden leer estos trabajos literarios. Además, la versión del poema es un factor importante. No es posible ejecutar la versión de un poema escrita en 1995 en una computadora de 2017. Para superar este problema, utilicé un simulador de computadoras antiguas que logra hacer funcionar el poema. He aquí

una captura de pantalla de este poema, consultado en mi computadora Linux, usando el simulador VirtualBox.

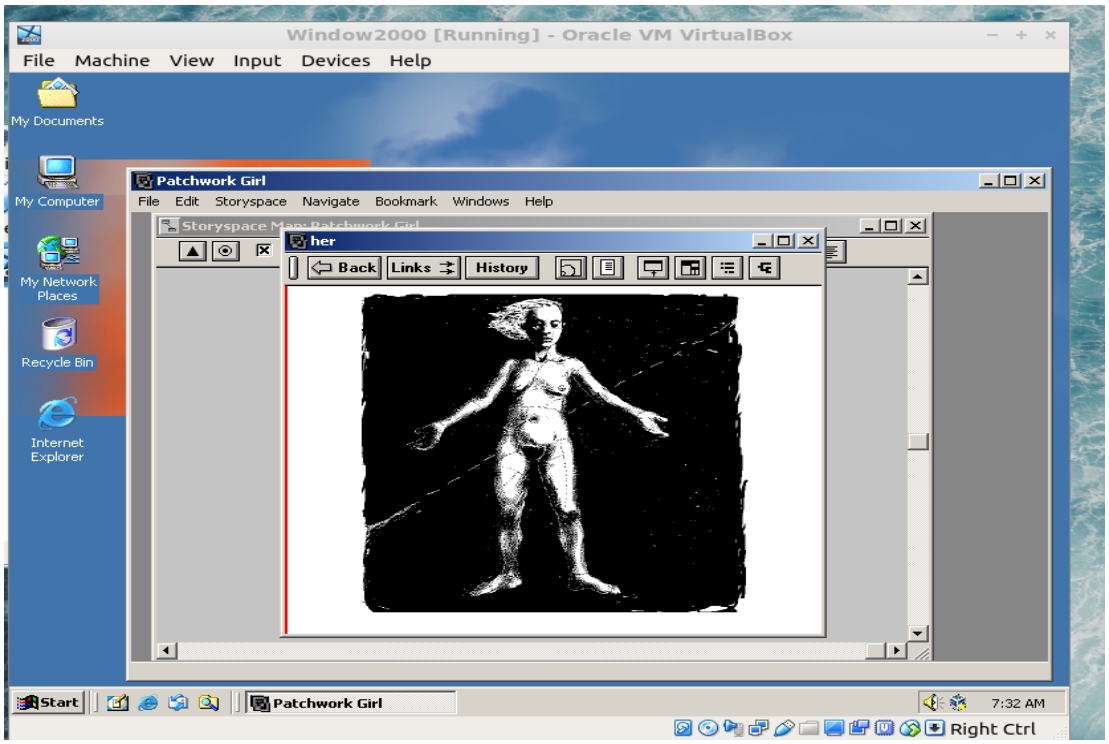

Fig. 1: Patchwork Girl de Shelley Jackson

Eastgate systems, en la última versión de su software, publicó una nueva versión de este trabajo para las nuevas computadoras Macintosh[3]. Si la empresa no hubiera proporcionado una nueva versión para computadoras recientes, este trabajo literario no habría sido accesible a un público general, que no es experto en informática. Sucede también que por razones similares hay muchos otros poemas digitales que ni siquiera los expertos pueden hacer funcionar hoy en día: están muertos.

Como antes expliqué, un poema digital y un poema impreso son ontológicamente diferentes. *Patchwork Girl* no es como *Trilce* de Vallejo. El

[3] Yo tenía una versión antigua de *Patchwork Girl* y para la nueva versión me obligaron a pagar 25 dólares como un filtro comercial que permite o no acceder al poema.

vocabulario que estoy discutiendo en este capítulo proporcionará un marco para entender la dinámica literaria de estos trabajos.

I-I-II

Artefactos cognitivos en acción: sistemas operativos

Una computadora moderna consta de uno o más procesadores, una memoria principal, discos, impresoras, un teclado, un ratón, una pantalla, interfaces de red y varios otros dispositivos de entrada/salida. En general, es un sistema complejo. Si cada programador tuviera que entender cómo funcionan todas estas cosas en detalle, acabaría sin poder escribir ningún código. Además, la gestión de todos estos componentes y su uso óptimo es un trabajo sumamente desafiante. Por esta razón, las computadoras están equipadas con una capa de software denominada sistema operativo, cuyo trabajo es proporcionar programas al usuario que siguen un modelo de la computadora mejorado, más simple, más limpio, y que le permiten manejar la gestión de todos los recursos que se acaba de mencionar. [Tanenbaum, 2014, pp. 1-2]

Hoy en día, la mayoría de la gente tiene un poco de experiencia con sistemas operativos como Windows, Linux, FreeBSD, o OS X, aunque las apariencias pueden ser engañosas. El programa con el que los usuarios interactúan, usualmente llamado Shell, cuando se basa en texto, y GUI (interfaz gráfica de usuario), cuando utiliza iconos, no es en realidad parte del sistema operativo, aunque utiliza el sistema operativo para hacer su trabajo.

Una descripción simple de los componentes principales se ofrece en la Fig. 2. Aquí vemos el hardware en la parte inferior. El hardware consiste en chips, tableros, discos, un teclado, un monitor, y objetos físicos similares. En la parte superior del

hardware está el software. La mayoría de las computadoras tienen dos modos de operación: modo kernel y modo usuario. El sistema operativo, pieza fundamental y principal de software, se ejecuta en modo kernel (también llamado modo supervisor). [Tanenbaum, 2014, p. 2]

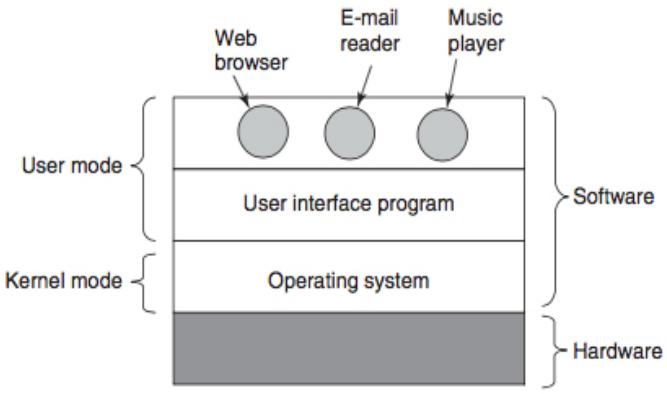

Fig. 2: componentes principales de sistemas operativos [Tanenbaum, 2014]

La diferencia principal de estos dos modos radica en su acceso al hardware. Cada hardware acepta algunas instrucciones y el modo kernel tiene acceso completo e irrestricto al hardware subyacente. Sin embargo, el modo usuario puede comunicarse con el hardware a través del modo kernel y no tiene acceso directo al hardware.

En las próximas discusiones sobre poesía digital, cuando me refiero al software, siempre hablo de programas relacionados con el modo usuario y cuando me refiero a sistemas operativos, hablo de programas que están relacionados con el modo kernel. Por ejemplo, los poemas hipertextuales escritos por StorySpace funcionan en modo usuario y se los puede utilizar en sistemas operativos como Dos y Mac que se

ejecutan en modo kernel. StorySpace, por razones comerciales, no proporcionó un software que pudiera ejecutarse en sistemas operativos gratuitos como Linux.

El desarrollo de este o aquel sistema operativo no es solo una cuestión técnica: también es una cuestión altamente política. Windows, Macintosh o Linux tienen diferentes kernel, diferentes programas y aspectos tecnológicos. Pero, al mismo tiempo, se han creado con diferentes ideologías políticas que definen sus límites. Por ejemplo, en 1983, Richard Stallman creó un sistema operativo de computadora tipo Unix compuesto enteramente de software libre. También lanzó el movimiento de software libre. Asimismo, Stallman fue pionero en el concepto de *copyleft*, que utiliza los principios de la ley de derechos de autor para preservar el derecho a usar, modificar y distribuir software libre. Puede parecer muy extravagante o ambiguo cuando Stallman dice: “el movimiento del software libre tiene mucho en común con el de Mahatma Gandhi”. Sin embargo, el movimiento de software libre ahora se considera un movimiento social y no sólo un movimiento tecnológico. Si nuestras ideas políticas tienen algún impacto en la lectura de un poema impreso de García Lorca, asesinado por Franco, o en un poema escrito por Manuel Machado, que elogió a Franco, cada lectura de un poema digital en un sistema operativo Linux o Windows también tiene repercusiones políticas de índole similar, pues conlleva otras consecuencias también políticas relativas al sistema operativo que utiliza. De manera semejante, un poeta-programador que elige producir sus poemas en Macintosh comparte una ideología política diferente a la de un poeta-programador que escribe sus obras en Windows o en Linux.

Además de la dimensión política, los sistemas operativos han configurado nuestra comprensión de los objetos digitales como las aplicaciones o los poemas digitales en un nivel cognitivo. La distinción analizada en esta sección sobre el modo de usuario y el modo de kernel es la principal distinción para definir la superficie y la profundidad en el entorno digital. La superficie de un poema digital reside donde ocurre la interacción con el usuario y la profundidad es donde el objeto digital interactúa con el sistema operativo y, por lo tanto, con el hardware. El sistema operativo Windows está diseñado con DOS como el kernel y, además, todos los demás software están desarrollados casi en un orden lineal. Windows es un collage de varios programas que interactúan al mismo nivel que otros y con el kernel. Sin embargo, el sistema operativo UNIX y otros que utilizan UNIX como su base por ejemplo el Mac o el Linux, se desarrollan con una estructura en capas. Es decir, como una cebolla, hay un núcleo mínimo y encima de ese otro nivel y la misma estructura continúa a la superficie. En ambas arquitecturas de sistemas operativos, lineales o en capas, la superficie del sistema operativo está dentro de los programas que interactúan con el usuario. Como ejemplo de cómo el diseño del sistema operativo influyó en la comprensión de los conceptos de profundidad y superficie en la poesía digital, menciono la tipología de Wardrip-Fruin para la comunicación textual. Este teórico es programador y la cultura del campo influye en su modelo teórico.

Wardrip-Fruin y Aarseth sugieren sus tipologías para la comunicación textual:

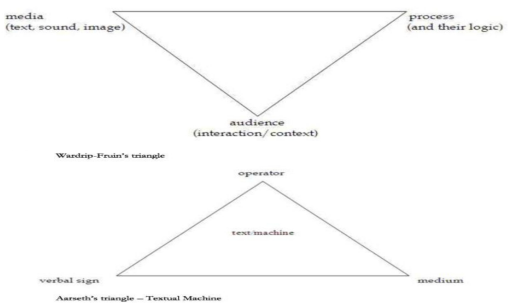

Fig. 3: Comparación de maquina textual de Aarseth y triangulo de Wardrip-Fruin [Wardrip-Fruin, 2009]

Ambos autores aunque utilizan terminologías diferentes reconocen las mismas entidades: operador o audiencia que es el lector o usuario de un texto digital, signo verbal o medio que es la materialidad de los signos textuales, y medio o proceso que es la estructura en la que se se producen los signos textuales.

Wardrip-Fruin escribe:

> All the works of digital literature are somehow presented to their audience – whatever on the teletypes, in web browser windows, through immersive installations, or by other means. If the audience is able to interact with the work, the means for this are also part of the work. I will call this site of presentation and possible (interaction) the work's surface. It may be as simple as a generic personal computer, consist of a large space or dizzying number of devices, or even take unexpected form (e.g, The Impermanence Agent makes all web browsing part of its interaction surface) [Wardrip-Fruin, 2009].

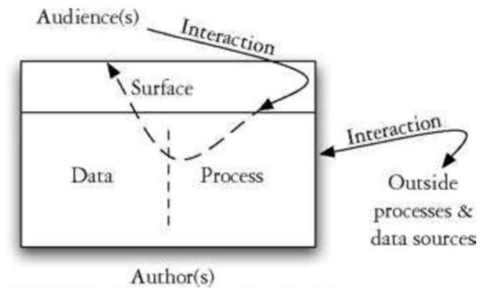

Fig. 4: modelo de literatura digital [Wardrip-Fruin, 2009]

La idea de superficie en su modelo corresponde con la distinción entre modo de usuario y modo de kernel en un sistema operativo.

I-I-III

Lenguajes de Programación

Para su ejecución real, los algoritmos, como conjuntos ordenados de operaciones para lograr un determinado fin, deben ser formalizados apropiadamente usando las construcciones proporcionadas por un lenguaje de programación. En otras palabras, los algoritmos que queremos ejecutar deben ser representados usando las instrucciones de un lenguaje de programación. Este lenguaje se definirá formalmente en términos de una sintaxis específica y una semántica precisa. Así, un lenguaje de programación es un formalismo artificial en el que se pueden expresar algoritmos. A pesar de toda su artificialidad, este formalismo sigue siendo un lenguaje.

Es evidente que un programa intérprete debe realizar las operaciones especificadas del idioma que está interpretando. Este intérprete, en los lenguajes de programación, simula una máquina virtual, donde el lenguaje de máquina es similar al lenguaje fuente. Sin embargo, a pesar de la diversidad de los lenguajes que lo anterior plantea, es posible discernir los tipos de operación y un “método de ejecución” común a todos los intérpretes. Según Gabbrielli y Martini:

> The type of operation executed by the interpreter and associated data structures, fall into the following categories:
>
> 1. Operations for processing primitive data;
> 2. Operations and data structures for controlling the sequence of execution of operations;
> 3. Operations and data structures for controlling data transfers;

4. Operations and data structures for memory management.
[Gabbrielli, 2010, p. 3]

Como se puede apreciar, esta clasificación corresponde, grosso modo, con la manera en que la máquina gestiona su trabajo para producir los resultados que se le solicitan, pero es hasta cierto punto irrelevante para un usuario común.

Los programas y aplicaciones escritos en un lenguaje de bajo nivel son ejecutables directamente en el hardware sin ninguna traducción. Llamamos de ‘bajo nivel’ a los idiomas que están más cerca del hardware. En otras palabras, su función principal es operar, administrar y manipular el hardware y sus componentes. El lenguaje de máquina y el lenguaje ensamblador son ejemplos populares de lenguajes de bajo nivel. Es, por ejemplo, el lenguaje que utiliza una impresora sin que nos demos cuenta ni tengamos ningún acceso directo a él.

A partir de finales de la década de 1940, estos lenguajes fueron usados para programar las primeras computadoras pero resultaron ser extremadamente torpes de utilizar. Debido a que las instrucciones en estos idiomas tenían que tener en cuenta las características físicas de la máquina, las cuestiones que eran completamente irrelevantes para el algoritmo tenían que ser consideradas al escribir los programas, o incluidas en algoritmos de codificación. Hay que recordar que a menudo cuando hablamos genéricamente de “lenguaje de máquina”, nos referimos a la lengua de una máquina física. Un lenguaje particular de bajo nivel para una máquina física es su lenguaje de ensamblaje, que es una versión simbólica de la máquina física; es decir, que utiliza símbolos como ADD para añadir, MUL para multiplicar, etc., en lugar de

sus códigos binarios de hardware). Los programas en lenguaje ensamblador se traducen al código de máquina utilizando un programa llamado ensamblador.

Por ejemplo, el siguiente código en lenguaje de ensamblador calcula el máximo común divisor (MCD) de dos o más números enteros que por definición es el mayor número entero que los divide sin dejar residuo:

```
mcd:
cmp r1, r2 @; r1 - r2 set cpsr
subgt r1, r1, r2 @; sub if r1 > r2 and put result in r1
sublt r2, r2, r1 @; else sub if r2 < r1 and put result in r2
bne mcd
```

Por otro lado, los lenguajes de programación de ‘alto nivel’ son los que apoyan el uso de construcciones que utilizan mecanismos apropiados de abstracción para asegurar que sean independientes de las características físicas de la computadora, el hardware. De esta forma, los idiomas de alto nivel son adecuados para expresar algoritmos de maneras relativamente fáciles de entender para el usuario humano. Sin embargo, incluso las construcciones de un lenguaje de alto nivel deben corresponder a las instrucciones de la máquina física, porque deben hacer que sea posible ejecutar los programas.

Los idiomas de alto nivel siempre han sido diseñados con el objetivo de ayudar a la tarea de programar computadoras. Sin embargo, desde los años 50 hasta el presente, la importancia y el costo de los diversos componentes involucrados en la implementación de los programas ha cambiado y por lo tanto las prioridades al diseñar un idioma también se han transformado por completo.

En los años 50, el hardware era sin duda el recurso más costoso e importante. Por ejemplo, Thomas Watson, Presidente de IBM, afirmó en 1943 que habría un mercado mundial para 5 computadoras. Por lo tanto, los primeros idiomas de alto nivel fueron diseñados con el objetivo de obtener programas eficientes que utilizaran al máximo el potencial del hardware. Esta situación en que se encontraron los primeros lenguajes se reflejó en la presencia de muchas construcciones que fueron inspiradas directamente por la estructura de la máquina física. El hecho de que la programación fuera muy difícil y requiriera tiempos muy largos se consideró un problema de importancia secundaria, que podría resolverse mediante grandes cantidades de recursos humanos, que ciertamente eran menos costosos que el hardware.

Hoy, el panorama es el opuesto. El hardware es relativamente barato y eficiente y los costos preponderantes del desarrollo del sistema de información están relacionados con las tareas que realizan los especialistas en informática. Por lo tanto, los lenguajes modernos se diseñan teniendo en cuenta primero el mejoramiento de las diversas actividades de proyectos de software, mientras que la preocupación por el uso eficiente de la máquina física han descendido a un lugar secundario, salvo en algunos casos particulares. [Gabbrielli, 2010, p. 22]

En un lenguaje de alto nivel, como Java, el mismo código que calcula MCD, se escribe así:

```
public class MCD {
   public static void main(String[] args) {
      int n1 = 81, n2 = 153, mcd = 1;
      for(int i = 1; i <= n1 && i <= n2; ++i)
```

```
        {
           // Checks if i is factor of both integers
           if(n1 % i==0 && n2 % i==0)
              mcd = i;
        }
        System.out.printf("M.C.D of %d and %d is %d", n1, n2, mcd);
     }
}
 }
```

El número de palabras que comparte este código fuente con un lenguaje natural como el inglés es mayor que el número de palabras utilizadas del inglés en el código fuente escrito en el lenguaje ensamblador. Además, por ejemplo, en lenguaje ensamblador, el acto de imprimir el resultado se muestra al poner un valor en una variable llamada r2; sin embargo, imprimir el resultado en Java se expresa de una manera más comprensible para humanos: System.out.printf. Es decir, el programa ordena al sistema que imprima el resultado.

Durante décadas, los poetas digitales utilizaron una variedad de lenguajes y entornos de computación. Los poetas de hipertexto, que utilizaron StorySpace, evitaron involucrarse en programación. Otros poetas de hipertexto usaron un lenguaje simple como Html para presentar su trabajo. La poesía generativa usa lenguajes de programación de alto nivel, desde JavaScript hasta C o Python. La poesía de Twitter usa Python, Rubby u otros lenguajes de codificación. La manera de selecionar un lenguaje de programación para realizar un poema digital, parcialmente revela algunas de las estrategias creativas del poeta-programador. Por ejemplo, los poetas que están usando un entorno como Storyspace, no quieren preocuparse por las dificultades de programación y prefieren concentrarse en la planificación del trabajo final y su

estructura. Los poetas que usan programación complicada, asumen la codificación y sus desafíos como parte de su proceso creativo. Storyspace y otros programas para la creación de hipertexto, como HyperDyne o Twine, se concentran en la estructura del trabajo y su entorno maneja la mayor parte de la programación. Por ejemplo, *Metamorphoses* de Ovidio se ha presentado en una versión de hipertexto escrito en StorySpace. A continuación, aparece una captura de pantalla de la estructura de este "libro".

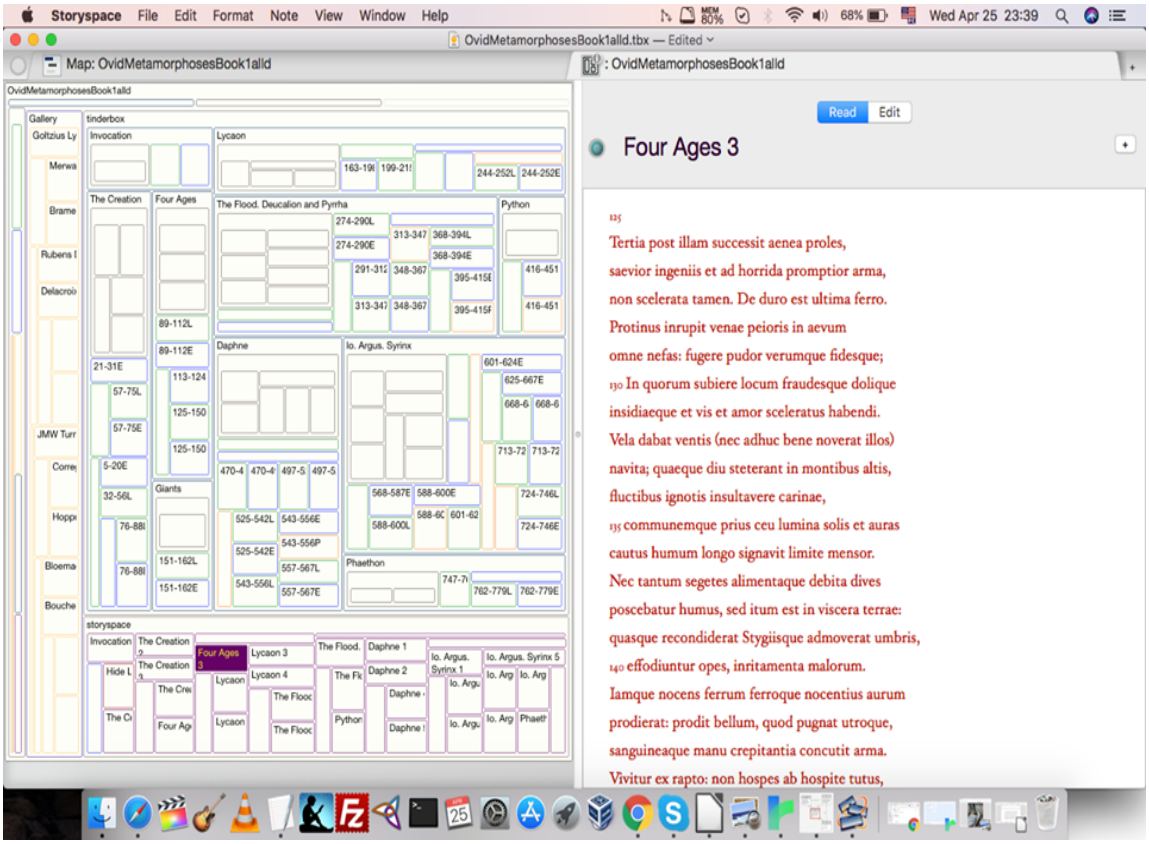

Fig. 5: la estructura hipertextual de Metamorphoses de Ovidio

Como se puede apreciar, la estructura visual está compuesta por dos partes. En la parte izquierda se observan celdas que indican el orden y la estructura de cada

página. Desde una página podemos llegar a algunas otras páginas definidas. Es decir, nuestro acceso a las páginas en el texto no es lineal. En la parte derecha se encuentra el texto con los enlaces a otras páginas del texto en un orden no lineal y también un texto legible con los aspectos visuales definidos en cada celda de la parte izquierda.

El caso de *Patchwork Girl* que mencioné anteriormente tiene una estructura como ésta:

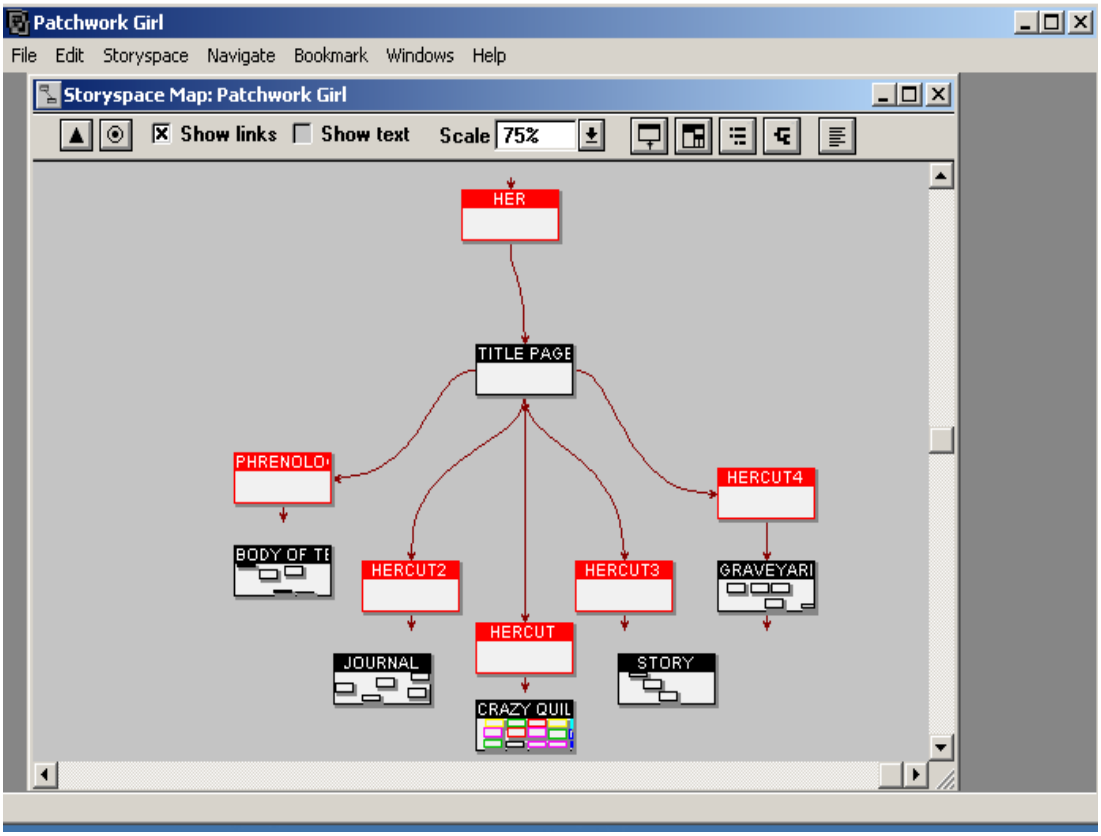

Fig. 6: la estructura de Patchwork Girl

Cada rectángulo define un estado que puede ser un capítulo o una página y el mapa completo con sus flechas y colores define a dónde podemos llegar desde una página o un capítulo.

Un generador de poemas como Jgnoetry, que puede producir poemas en diferentes formas, como un haiku o un tanka, esta escrito en JavaScript y HTML. Estos son algunos de los lenguajes de programación más fáciles de aprender para programadores que tienen cierta curiosidad por la programación, pero tienen miedo de utilizar lenguajes más complicados como Java o C que llevan años de experiencia en programación. Veamos una captura de la página de presentación de Jgnoetry:

Fig. 7: poesía generada por Jgnoetry

Esta página nos da la posibilidad de usar el botón generar / regenerar cada vez que queremos producir un nuevo poema. Podemos seleccionar y copiar el poema generado para nuestro uso en otros programas como Word, o podemos exportar el texto del poema a un archivo de texto para futuros usos, especialmente si queremos modificarlo por nuestra cuenta. La forma del poema puede determinarse seleccionando entre haiku o sonetos. También podemos ver las plantillas y los corpus utilizados para generar el poema. Asimismo es posible personalizar y cambiar las fuentes y algunos otros atributos visuales del poema usando el botón de opciones.

El siguiente es otro ejemplo de programación en código Java para Jgnoetry en que se definen las formas de estrofa del soneto y el haikú:

```
class SonnetStanza(Poem):
   def __init__(self):
      Poem.__init__(self, "Sonnet Stanza", magic="ABAB")

class Sonnet(Poem):
   def __init__(self):
      Poem.__init__(self, "Sonnet", magic="ABAB CDCD EFEF GG")

class Haiku(Poem):
   def __init__(self):
      Poem.__init__(self, "Haiku", magic="575")
```

La habilidad del poeta-programador en los lenguajes de programación no habla de la calidad de su trabajo poético, pero ciertamente demuestra su espíritu juguetón y su implicación con la máquina y también imprime rasgos singulares a su trabajo. Por otro lado, esto puede tener consecuencias negativas: el trabajo depende principalmente del programador y si el programador se niega a actualizar el trabajo para sistemas nuevos, tarde o temprano el trabajo morirá. Además, en muchos casos, la instalación y el uso del poema escrito en lenguajes complicados en un dispositivo digital requerirán más conocimientos técnicos que los trabajos escritos en lenguajes populares de alto nivel.

*****

Hasta aquí solo he presentado una breve semblanza de los principales movimientos en el proceso de desarrollo de los sistemas operativos y los lenguajes computacionales. La historia comienza con la Segunda Guerra Mundial y continúa con los grandes consorcios industriales y comerciales, algunos de ellos ahora

conocidos por ser monopolios en el mundo digital. En las primeras computadoras de segunda y tercera generación, el software era libre y abierto. Incluso en la década de 1970, AT&T distribuyó las primeras versiones de UNIX sin costo alguno para el gobierno y los investigadores académicos. Sin embargo, después de la cuarta generación, el costo creciente del desarrollo del software introdujo el concepto del programa como producto y muchas empresas comenzaron a cobrar por el software. En 1983, como ya dije anteriormente, Richard Stallman lanzó el proyecto GNU para escribir un sistema operativo completamente libre. En 1989, se publicó la primera versión de la Licencia Pública General de GNU. El núcleo de Linux, iniciado por Linus Torvalds, fue lanzado como un código fuente libremente modificable, en 1991.

Hoy en día hay una tensión entre la concesión monopolista de derechos de propiedad de la información y las necesidades democráticas para ampliar el flujo y el acceso a esta información. Derechos de autor y otras leyes de propiedad intelectual buscan restringir el acceso para que sólo los capaces y dispuestos a pagar una determinada suma puedan hacer uso de la obra. Por lo tanto, esta restricción de acceso puede reducir realmente la capacidad de algunos miembros de la sociedad de obtener la información que necesitan para tomar decisiones sociales, económicas y políticas fundamentadas. De esta manera, la brecha entre una información “rica” y una “pobre” se ensancha. Las acciones del movimiento de código abierto y software libre, que se basan en compartir el código estructural y el contenido que se encuentra en él (es decir, los algoritmos y el sentido del código), entran directamente en conflicto con los intereses de los dueños de derechos de autor y otros derechos de

propiedad intelectual Esto sucede porque si el código es abierto, no hay ninguna manera en la que los métodos de protección, comúnmente conocidos como administración de derechos digitales, que sirven como las cerraduras en los trabajos creativos, puedan mantenerse en secreto. Parte de la política de los nuevos medios se encuentra en esta tensión. Voy a explicar tal tensión y su impacto en la creatividad al detalle cuando aborde el dominio o campo de la poesía digital.

Entre 1945 y 2002, la computadora se transformó una y otra vez, cada vez redefiniendo su esencia. "La computadora" comenzó como una calculadora científica rápida; Eckert y Mauchly lo convirtieron en UNIVAC, una máquina para el procesamiento de datos en general. Ken Olsen la transformó en un procesador de información en tiempo real, que trabajó simbióticamente con sus usuarios. Ed Roberts la convirtió en un dispositivo que cualquiera podría poseer y usar. Steve Jobs y Steve Wozniak la cambiaron en un dispositivo que era útil y divertido. Gary Kildall y William Gates la transformaron en una plataforma estandarizada que podría ejecutar una cornucopia de software comercial vendido en tiendas minoristas. Bob Metcalfe, Tim Berners-Lee y otros la transfiguraron en una ventana hacia una red global. Cada modificación estuvo acompañada de afirmaciones de que era improbable que se produjeran más transformaciones; sin embargo, cada vez alguien más lograba abrirse paso.

****

A pesar de que los hardwares desempeñan un papel importante en la descripción de la tipología de artefactos cognitivos como las computadoras, no voy a mencionar aquí

la historia y el proceso de su desarrollo. Las cuatro funciones de cualquier artefacto cognitivo dependen tanto de su hardware como de su software. La velocidad con la que un programa puede almacenar y recuperar información en y desde la memoria depende del hardware, así como la capacidad de búsqueda o incluso el razonamiento. Los mencionaré cuando esté analizando los poemas digitales. Algunos poemas son específicos de determinado hardware mientras otros no lo son. Sin embargo, todos ellos necesitan una capacidad mínima de memoria de computadora.

I-I-IV

Metodologías de desarrollo de software

Jim Andrews, al comienzo de su código fuente para *Blue Hyacinth Stir Fry Text*, un poema de hipertexto presentado en 2004, escribe:

> <!--
> Thanks to Pauline Masurel for a great text and the blue squares and
> the text colors.
>
> Thanks also to Marko Niemi for showing me how to fix the code
> so it runs on many more browsers than the original version ran on.
> I wrote the original code in 1999 and it ran only on IE for the PC. Marko
> showed me in 2004 how to get it running on the Mac and PC on IE 4+,
> Netscape 7+, Firefox, and Safari. Haven't tested it on Opera.
>
> Jim Andrews
> November, 2004
> -->

Esta nota contiene la siguiente información:

1. El trabajo se basa en un texto escrito por Pauline Masurel. El autor del texto también ayudó al autor del poema digital en aspectos técnicos como la apariencia visual del poema-programa.

2. El trabajo tenía problemas en algunos navegadores como Firefox y Safari. Marko Niemi, otro poeta digital, ayudó en resolver problemas técnicos. Es decir, Andrews primero hizo un prototipo del poema-programa en 1999, que estaba funcionando en algunos navegadores, y luego, con la ayuda de otro poeta-programador, pudo hacer algunos otros prototipos y probarlos en sistemas nuevos y entonces llegar a una versión que funcionaba en la mayoría de los navegadores de su tiempo.

Esta información habla de un proyecto de trabajo en equipo y una metodología para desarrollar software llamada *Metodología de Prototipado*. Además, habla de algunos pasos para desarrollar el poema: investigación, planificación, diseño y prueba. Sin embargo, el autor no publica ninguna documentación sobre el proceso de desarrollo, aunque las notas que incluye en el código fuente nos ayudan a tener algunas ideas sobre estos procesos. También la nota mencionada revela que el poeta digital, en esta etapa de su experiencia, no era un programador profesional, aunque estaba aprendiendo a programar, practicando a la vez que escribía el poema digital.

De igual manera que cualquier texto literario involucra procesos como preescritura, redacción, revisión, edición y publicación, cualquier poema digital también se publica bajo algunos procesos similares que se denominan procesos y metodologías de desarrollo de software.

Una metodología de desarrollo de software es un conjunto de reglas y lineamientos que se utilizan en el proceso de investigación, planificación, diseño, desarrollo, pruebas, instalación y mantenimiento de un producto de software. La metodología también incluye valores fundamentales que son sostenidos por el equipo del proyecto y las herramientas utilizadas en el proceso de planificación, desarrollo e implementación.

La construcción de un producto de software es un proceso que consiste en varias etapas distintas. Cada etapa tiene sus propias entregas y está limitada por un marco de tiempo específico. Dependiendo del proyecto, ciertas etapas ganan peso

adicional en el esfuerzo general por implementar el producto de software. Las etapas son las siguientes:

1. La investigación es la etapa en la que el propietario del proyecto, el gestor del proyecto y el equipo del proyecto se reúnen e intercambian información.
2. La planificación es la etapa en la que se establecen todos los elementos para desarrollar el producto de software. La planificación comienza con la definición del flujo general de la aplicación. El siguiente paso consiste en desglosar el flujo en subensamblajes más pequeños y más fáciles de administrar. Para cada subconjunto se debe definir un conjunto completo de funcionalidades. Basándose en la funcionalidad requerida, se diseña una estructura de base de datos.
3. El diseño es la etapa en la que se crea el diseño de la aplicación. Las aplicaciones web y las aplicaciones móviles tienden a otorgar más importancia a la disposición que las aplicaciones de escritorio. Dependiendo de la naturaleza de los diseños pueden ir desde lo tosco y lo funcional hasta lo complejo y lo artístico.
4. El desarrollo es la etapa donde se escribe el código y la aplicación de software se construye realmente. La etapa de desarrollo comienza con la configuración del entorno de desarrollo y el entorno de prueba, los cuales deben sincronizarse utilizando siempre el mismo protocolo. El código se

escribe en el entorno de desarrollo y se carga en el entorno de prueba mediante el protocolo de sincronización.

5. La prueba es la etapa en la que se identifican y corrigen los errores de programación y diseño. Los errores de programación son escenarios donde la aplicación se bloquea o se comporta de una manera que no se suponía según la arquitectura diseñada.

6. La instalación es el escenario donde se instala la aplicación en el entorno en vivo. La etapa de configuración precede a la explotación real del producto de software. La configuración implica configurar el entorno en vivo en términos de seguridad, hardware y recursos de software.

7. El mantenimiento es la etapa que cubre el desarrollo de software posterior a la configuración de la aplicación y también es la etapa responsable de asegurar que la aplicación se ejecute dentro de los parámetros planificados.

Existen más de 30 paradigmas de desarrollo de software. Para tener una idea sobre los paradigmas voy a mencionar cuatro de ellos. Es importante considerar las etapas antes mencionadas. Todas ellas forman parte de cada metodología, pero el orden de cada una de ellas es variado. Es decir, cada metodología actúa como una gramática para el uso de las etapas. Por ejemplo, en las metodologías de Waterfall todas las etapas aparecen una tras otra, y en las etapas de la metodología de código abierto se descentralizan y se interconectan dinámicamente.

Waterfall o cascada es la primera metodología generalmente reconocida como metodología dedicada al desarrollo de software. La metodología de la cascada es un proceso lineal secuencial, donde cada etapa comienza sólo después de que la etapa anterior se ha completado. La metodología de cascada es predecible y valora rigurosamente la planificación y la arquitectura del software.

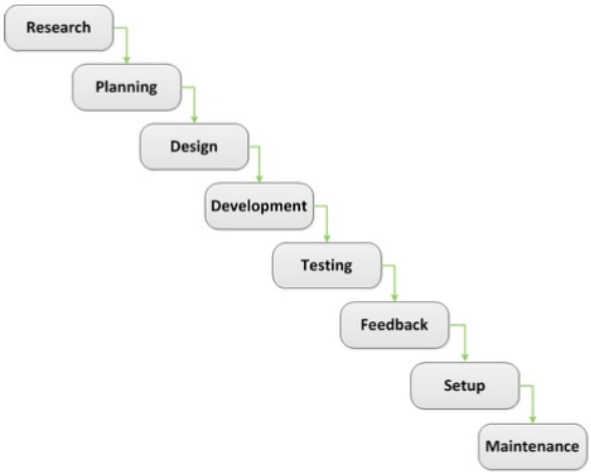

Fig. 8: Metodología Waterfall o Cascada [Despa, 2014]

Prototipado es una metodología que evolucionó de la necesidad de definir mejor las especificaciones y que conlleva la construcción de una versión demo del producto de software que incluye la funcionalidad principal.

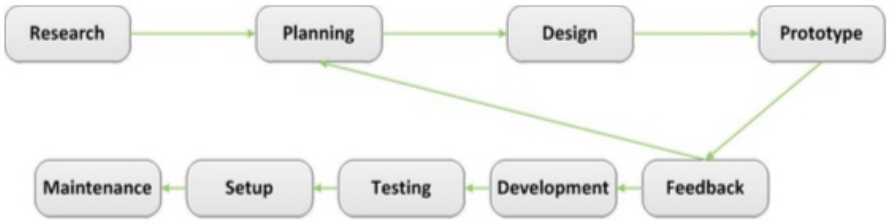

Fig. 9: Metodología Prototipado [Despa, 2014]

La Espiral es una metodología que se centra en la identificación de objetivos y el análisis de alternativas viables en un contexto bien documentado de las limitaciones de los proyectos. La metodología espiral tiene 4 fases principales: planificación, análisis de riesgos, desarrollo y evaluación. El proyecto seguirá cada fase varias veces en el orden antes mencionado hasta que la aplicación de software esté lista para ser configurada en el entorno en vivo.

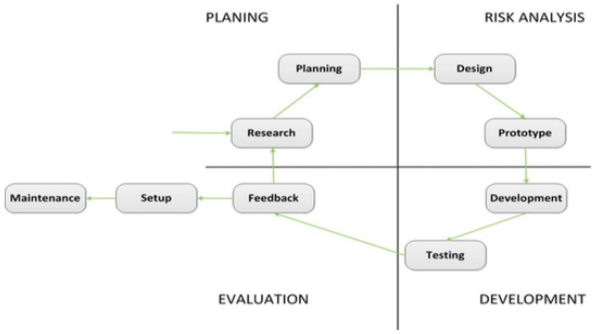

Fig. 10: Metodología Espiral [Despa, 2014]

El desarrollo de software de código abierto es una metodología descentralizada sin autoridad central, dueño del proyecto, sin compensación para el equipo del proyecto, sin responsabilidad y sin embargo con una alta tasa de éxito.

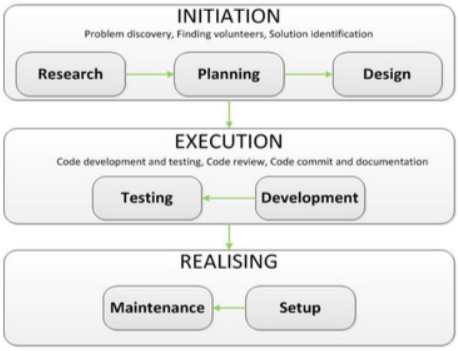

Fig. 11: Metodología Código Abierto [Despa, 2014]

Una vez que comentamos rápidamente las metodologías de desarrollo de software, pasemos a los paradigmas de programación, que son una forma de clasificar los lenguajes de programación basándose en sus características. Los lenguajes pueden clasificarse en múltiples paradigmas.

Los paradigmas de programación de nivel más bajo consisten del código de máquina, que representa directamente las instrucciones (el contenido de la memoria del programa) como una secuencia de números, y el lenguaje ensamblador donde las instrucciones de la máquina están representadas por mnemotecnias y las direcciones de memoria pueden recibir etiquetas simbólicas. A veces se les llama lenguajes de primera y segunda generación.

El siguiente avance fue el desarrollo de lenguajes procesales. Es decir, lenguajes que describen, paso a paso, exactamente el procedimiento que debe, según el programador, ser seguido para resolver un problema específico. La eficacia de cualquier solución de este tipo es, por lo tanto, totalmente subjetiva y altamente dependiente de la experiencia, inventiva y habilidad del programador.

Tras el uso generalizado de lenguajes procesales, se crearon lenguajes de programación orientados a objetos (OOP). En estos lenguajes, los datos y métodos para manipularlos se mantienen como una unidad llamada objeto. La única forma en que otro objeto o usuario puede acceder a los datos es mediante los métodos del objeto. Por lo tanto, el funcionamiento interno de un objeto puede cambiarse sin afectar a ningún código que utilice el objeto.

***

Para dar un ejemplo de las terminologías introducidas, trataré de explicar los conceptos mencionados para un generador de poesía del que ya hablamos: jGnoetry. Supongamos que recibimos una oferta para escribir un programa que genere poesía. En todas las metodologías mencionadas, el primer paso para desarrollar un software de este tipo es la investigación en la que el programador/poeta buscará las soluciones existentes. También el poeta/programador debe considerar qué se puede implementar en un período de tiempo determinado. Por lo tanto, el presupuesto, el equipo de desarrollo y los factores de riesgo deben tenerse en cuenta. Finalmente, elegirá la metodología a emplear dependiendo de la complejidad de la tarea y del número de participantes y su organización.

Gnoetry Daily es un sitio web dedicado a las experiencias poéticas que utilizan programas como jGnoetry. El sitio web define sus objetivos de la siguiente manera:

> To some extent we explore methodologies as OULIPO did, but we don't mind randomness or surrealists [sic], and often we also want to produce expressive poems (or at least poems that are open to a multiplicity of readings.) Conceptualists talk about how computation and data has affected poetry, but we just take that as obvious. Some of our poems produce unusual and irreverent juxtapositions like the Flarf poets, but that's just one possibility we explore. Digital Poets tend to create generative interfaces whose purpose is to make a statement to a general audience, while our generators tend to be idiosyncratic tools focused on our authoring needs, and our focus is on the poems output rather than on the generators.

OULIPO, que se presentará más adelante en mi recuento de la poesía digital, introdujo algunos aspectos clave de la poesía generativa. El Ouvroir de Litterature Potentielle (OULIPO) o Taller de literatura potencial, fue fundado en 1960 por el matemático francés François Le Lionnais y el escritor Raymond Queneau. El objetivo era investigar las posibilidades del verso escrito bajo un sistema de limitaciones estructurales. La filosofía de OULIPO busca conectar la poesía y las matemáticas. Lionnais y Queneau creían en el potencial profundo de un poema producido dentro de un marco o fórmula, el cual, si se realiza adoptando una postura lúdica, genera resultados que podrían ser infinitos. Las ideas de las limitaciones estructurales son mas antiguas que las de OULIPO. Por ejemplo, la rima y la métrica son ejemplos de estas restricciones. Algunas de las limitaciones estructurales de OULIPO más populares se mencionan aquí:

S + 7: se reemplaza cada sustantivo en un texto con el séptimo sustantivo después de él en un diccionario.

Lipograma: escrito que excluye una o más letras.

Univocalismo: poema que usa sólo una vocal.

Palíndromos: sonetos y otros poemas construidos de tal manera que se leen igual en un sentido y su contrario, como ocurre con la frase “dábale arroz a la zorra el abad”, en que la inversión es a nivel de las letras.

jGnoetry o en general Gnoetry (Poesía Generativa) utiliza la generación y la regeneración interactiva de poemas a partir de un modelo de lenguaje, *n-gram* (que explicaré a seguir), construido sobre textos fuente preexistentes, como novelas, poemas o artículos periodísticos, lo que en sí es una restricción como las propuestas por OULIPO.

En los campos de la lingüística computacional y la probabilidad, un *n-gram* es una secuencia contigua de n elementos de una muestra dada de texto o discurso. Los elementos pueden ser fonemas, sílabas, letras o palabras, según la aplicación. Los *n-grams* normalmente se recopilan a partir de un corpus de texto o de sonido.

De manera simplificada, jGnoetry y otros generadores de poemas similares, utilizan un corpus existente o una base de datos de textos y aplican algunas técnicas computacionales para manipularlos. Es decir, usan una forma poética -soneto o haiku, etc.-, una técnica de manipulación textual -aleatoriedad, restricciones tipo OULIPO, etc.- y un corpus para producir un nuevo texto poético. Se puede decir que los poemas generativos son el resultado de una intertextualidad deformada o manipulada.

Carolyn Lamb reconoce cinco métodos para producir poesía generativa:

1. Plantillas. La generación de plantillas es uno de los medios más simples para construir poemas. La generación de plantillas, también llamada llenado de espacios, ha sido común desde el primer programa de poesía generativa, "Textos estocásticos", de Theo Lutz. Lamb explica:

> The basic steps are as follows:
> 1. Create lists of words or phrases in different categories, e.g. nouns or verbs.
> 2. Create one or more line templates with slots into which a word from a given list can be inserted.
> 3. Randomly select a word from the appropriate list to fill each slot.

2. Markov encadenando. Una cadena de Markov es un modelo estadístico aplicado a los datos de una serie. Es similar a las técnicas de *n-grams* que mencioné anteriormente, con la diferencia de que obedece reglas en que, dado un conjunto de condiciones de entrada, se producen, dependiendo de cada caso, distintas respuestas, las cuales a su vez, son condiciones de entrada, para el siguiente proceso, y así sucesivamente.

3. Gramáticas libres de contexto. Es una forma de generalizar el método de la plantilla. En lugar de tener sólo una plantilla, el programa define una simple gramática. Por ejemplo, un espacio [SINTAGMA-NOMINAL] podría llenarse con un sustantivo, pero también con "el [SUSTANTIVO ADJETIVO]", "el [SUSTANTIVO] que [SINTAGMA-VERBAL]", o una variedad de otras formas gramaticales que cumplan la función requerida.

4. Poesía encontrada. Otro método es omitir el proceso de generación y, en su lugar, usar una computadora para recolectar texto escrito por humanos. Cualquier texto, desde noticias hasta el resultado de una búsqueda en Google, se puede usar aquí. Leevi Lehto, el famoso poeta finlandés, utilizó esta técnica para su generador de poesía de Google y más tarde publicó un libro impreso de poesía modificando los textos generados.

5. Métodos diversos. Hay algunos otros métodos para modificar un texto. Por ejemplo, permutando las palabras en una frase corta. Sin embargo, no son muy comunes.

Después de la investigación, comenzará la planificación. En nuestro caso, los datos provienen de diferentes fuentes: textos existentes e interacción humana. Además, un poema que adopte muchas formas diferentes y que tenga un solo módulo para implementarlas todas puede ser imposible dada la fecha límite. Por lo tanto, el poeta/programador selecciona un grupo de formas que podrían implementarse en el lapso de tiempo dado. En este caso los poetas/programadores eligieron las formas Haiku, Tanka, Renga, Blank Verse y Syllabic. Los corpus de texto que son la fuente de datos de los poemas también son elecciones importantes, que pueden corresponder a autores como Julio Verne o Joseph Conrad. Se habrá de decidir también si el programa será funcional en todos los sistemas operativos o no, si será para PC o teléfonos móviles o ambos. En este caso la aplicación fue diseñada en Python, lenguaje que puede ser utilizado en Windows, Linux y Mac OS. No es una aplicación para celular.

La siguiente imagen explica cómo funciona jGnoetry para los poemas generados en Gnoetry Daily[4]:

[4]https://gnoetrydaily.wordpress.com/2018/03/25/comments-on-a-taxonomy-of-generative-poetry-techniques/

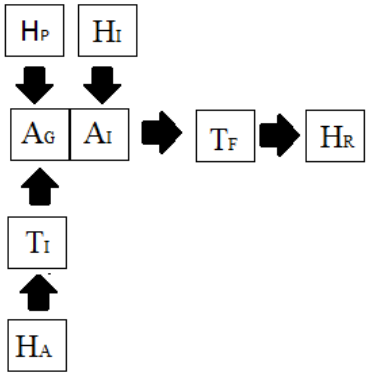

Fig. 12: la maquina textual de jGnoetry

Donde:

Hp: el programador humano (autor)
Ag: el generador algorítmico (jGnoetry)
Ai: la interfaz algorítmica (jGnoetry)
Ti: el texto de entrada
Ha: el autor humano del texto de entrada
Hi: el usuario de la interfaz humana
Tf: el texto final
Hr: el lector humano

Después de la planificación, los poetas/programadores comienzan a diseñar el software del poema. Ellos diseñan la interfaz de usuario, la estructura de la base de datos, el algoritmo para cada módulo o forma de un poema.

Las siguientes imágenes muestran cómo funciona el programa original[5]:

1. Seleccionar un formulario, o una plantilla para el poema:

[5]https://mchainpoetics.wordpress.com/the-programs-info-and-screenshots/

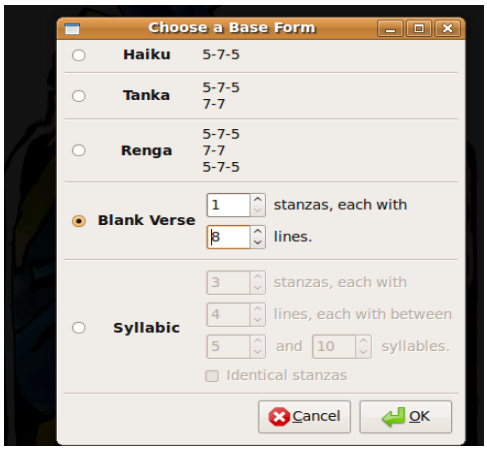

Fig. 13: elegir forma de poema

2. Seleccionar el corpus o seleccionar las intertextualidades:

Fig. 14: elegir el corpus

3. Seleccionar el porcentaje de cada corpus. El programa manipulará el corpus después de este paso.

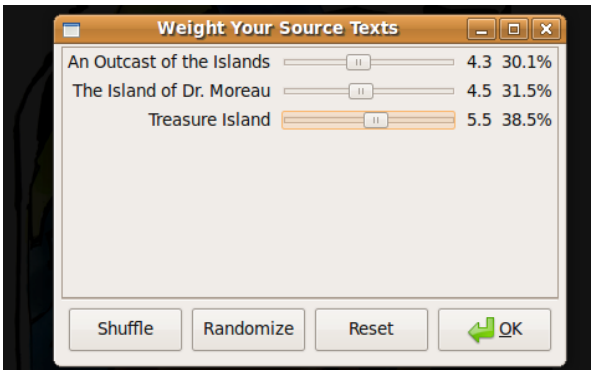

Fig. 15: seleccionar porcentaje de cada corpus

4. Resultado del programa o texto generado:

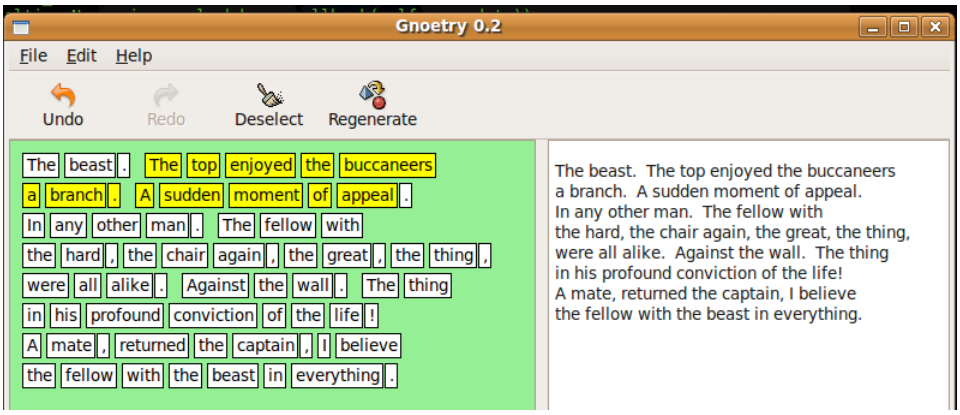

Fig. 16: poema generado por jGnoetry

Ahora comienza la implementación o desarrollo. Los programadores escribieron el código fuente de cada módulo y los probaron en sus computadoras. Compartieron el código fuente con otros desarrolladores para recibir comentarios. El lanzamiento ocurre cuando el programa está funcionando de acuerdo con el diseño. Los autores subieron el código fuente en línea para que la gente pudiera usarlo con determinadas condiciones.

En una breve nota dentro de uno de los códigos fuente del programa aparece esta información:

```
/*
   Copyright 2011 Edde Addad
   Based on Gnoetry by Jon Trowbridge and Eric Elshtain
```

```
    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
*/
```

La licencia utilizada aquí, GNU, habla de la metodología de desarrollo de software original de este programa: metodología de código abierto. Además, este texto explica que Edde Addad utilizó el software actual Gnoetry, que fue escrito originalmente por el lenguaje de programación de alto nivel de Python y se suspendió después de 2012. Carolyn Lamb describe la metodología utilizada por este programa y otro software relacionado como mera generación. Según ella, un problema en la poesía de la plantilla es la repetibilidad. A menudo, ejecutar un programa de plantilla varias veces produce un efecto repetitivo en el que la estructura de la plantilla se vuelve obvia. Sin embargo, la poesía generativa utiliza otras técnicas para evitar tales problemas: mejora la interferencia humana y la de la computadora. En la primera técnica, un ser humano edita el texto generado y, en la segunda, algunos algoritmos computacionales, como los algoritmos genéticos, se aplican para revisar el texto.

I-I-V

Literatura electrónica vs. juegos de computadoras

Como hemos dicho, la poesía, en su forma tradicional, no puede nunca tomar la forma de un videojuego porque los videojuegos, como los conocemos, en su forma popular (es decir, un conjunto de acciones rápidas, a las cuales el jugador responde físicamente), son la antítesis de los propósitos de un determinado estilo de poema.

Funkhouser explica:

> A large audience might consume a technologically complex digital poem produced as a video game, but that text is going to be vastly different from something in the anthologies heretofore published by W. W. Norton. Given a new set of stimuli, —a slower pace of presentation, materials absorbed as words and artwork— the typical video-game audience might change its tastes, but I do not see those radically different modes ever conjoining in titles that reach a high level of popularity in mass culture. Poets will build poetry-based games, I am sure —perhaps they will allow for real-time encounters with texts, possibly in multiuser interactive environments— yet their scale and purposes will differ vastly from what is available in arcades. These titles would be an educational tool and may have an influence on the circulation of ideas and level of visibility of conventional poetic works; their production should be encouraged. [Funkhouser, 2007, pp. 239-240]

Sin embargo, los videojuegos, y en general los juegos de computadora, tienen mucho en común con la poesía digital. La estructura cinemática, la interactividad, la estructura hipertextual, la interfaz y muchos otros elementos explicados principalmente por Manovich, Aarseth y Landow, están presentes en ambos. Los

conceptos introducidos por estos autores han sido utilizados con frecuencia en la lectura de los videojuegos y de la literatura digital.

Según Mark Overmars, profesor de programación de videojuegos en Universidad de Utrech, los cambios en los últimos cincuenta años en los juegos de computadora se pueden catalogar así:

- Cambios en el hardware para juegos
- Cambios en los dispositivos de interacción
- Cambios en las herramientas de software disponibles.
- Cambios en el negocio de los juego.
- Cambios en la demografía de los jugadores.
- Diversificación: inicialmente, los juegos de computadora fueron jugados sobre todo en máquinas de arcade. Hoy en día jugamos juegos en consolas, computadoras personales, dispositivos portátiles, teléfonos, etc.
- Cambios en el diseño de los juegos. [Overmars, 2012, p. 1]

Los juegos de computadora requieren tecnología capaz de manejar grandes cantidades de datos y de representaciones de estos datos. La relación entre un fenómeno tecnológico, como las computadoras, y la cultura no es una relación simple. Como hemos dicho, la tecnología puede inspirar (o habilitar) desarrollos culturales y los desarrollos culturales pueden inspirar nuevas tecnologías. Para citar un ejemplo obvio, los juegos de computadora fueron desarrollados originalmente en los equipos diseñados para fines militares y académicos. Pero hoy los juegos de computadora son la fuerza que impulsa el desarrollo de mucho hardware, tales como los aceleradores de gráficos tridimensionales.

También, la historia de los juegos de computadora de alguna manera explica la historia de los sistemas informáticos. A pesar de todas las similitudes que

comparten con la literatura digital, los juegos pudieron desarrollarse como una industria mucho más exitosa. Hoy en día, programadores, desarrolladores, compañías y escritores creativos participan en ella. En cambio, la literatura digital no compartió tal fortuna y todavía lucha por ser reconocida como literatura en muchas escuelas tradicionales. Las razones detrás de esto probablemente se encuentran en la diferente naturaleza del juego y de la literatura: el juego está diseñado para ser popular y la literatura busca subjetividades críticas, no necesariamente emparentadas con la popularidad. Sin embargo, tanto los juegos de computadora como los poemas digitales son software y usan tecnologías similares.

I-II

Segunda configuración: la poesía digital

Chris Funkhouser demuestra que los fundamentos de la poesía digital, mecánicamente y conceptualmente construidos en las décadas anteriores a las computadoras personales, fueron establecidos firmemente en la década de 1990, antes de que la WWW empezara a existir de manera pública.

Según Funkhouser, la poesía digital no es un género singular o "forma", sino más bien un conglomerado de formas que ahora constituye un género, a pesar de que la propia actividad creativa — en términos de sus medios, métodos e intenciones expresivas — contiene componentes heterogéneos. En *Prehistoric Digital Literature* Funkhouser observa:

> Poets initially used computer programs by synthesizing a database and a series of instructions to establish a work's content and shape. By the mid-1960s, graphical and kinetic components emerged, rendering shaped language as poems on screens and as printouts. Since then, video-graphic and other types of kinetic poems have been produced using digital tools and techniques. Beginning in the 1980s, hypertext (nonlinear texts that are intrinsically, mechanically interconnected) developed in sync with the in-creasing availability of the personal computer. A few other experimental forms, like audio poetry, appeared along with new technical advancements. When the WWW emerged, multimedia, transcontinental, hyperlinking poets began to spark expression through interconnected motherboards; the status of the art form has risen with the increasing affordability of computing and capabilities of network technologies. Only a few works of digital poetry titles are now circulated offline (few people are publishing digital poetry on diskette or even CD-ROM). The copious amount of material delivered to readers through the WWW is strong evidence that computers and telecommunications networks heighten the

> audibility and visibility of this strand of contemporary poetry. [Funkhouser, 2007, p. 2]

La poesía digital utiliza múltiples formas expresivas que vinieron antes de la WWW. El auge de la WWW no ha creado una ruptura entre el pasado y el presente, sino que representa otra etapa de avance tecnológico para la cultura en general y un momento de estabilización para la poesía digital. Giovanna di Rosario distingue cinco tipos de poesía digital que resumo a continuación:

1. E-Poetry cuya forma de expresión se basa en la noción de «segmento», es decir, en elementos morfológicos sin un reloj interior. Según Di Rosario "Segments are morphological elements without an inner clock, such as text blocks, images, and so on." [Di Rosario, 2011, p. 105] Posteriormente añade: "In kinetic works poems are transformed under the reader eyes due to movement of and within the text." [Di Rosario, 2011, p. 106] El segmento puede ser un solo bloque de texto, o muchos segmentos pueden trabajar juntos.
2. E-Poetry cuya forma de expresión se basa en la noción de la «secuencia», es decir, que tiene una temporalidad interior, un reloj interior. Di Rosario intenta no utilizar la terminología ya empleada para clasificar esta categoría de textos, normalmente descritos como obras cinéticas (u obras visuales y cinéticas), sino que se refiere a la terminología semiótica. Di Rosario explica: "It is true that sequence-based e-poems are kinetic: sequences are kinetic,

however, also segments-based e-poetry can be kinetic in some ways. The reader can give movement to a text with her interaction. What I would like to differentiate in this typology is the nature of time." [Di Rosario, 2011, p. 146]. Cinético, aquí, significa caracterizado por el movimiento. El movimiento es una de las características de esta categoría de textos. En los trabajos cinéticos, la mutación óptica del texto, de las palabras, y de las letras, es su característica operativa principal. Los poemas se mueven y cambian ante los ojos del espectador. El movimiento sugiere nuevos significados para el lector e introduce nuevas figuras y tropos. La propuesta de esta categoría impone un tiempo de lectura sobre el texto, que marca la diferencia con la poesía electrónica basada en segmentos.

3. E-Poetry híbrida cuya forma de expresión se basa en las nociones de «segmento», «secuencia» e «hipertexto». Estos textos se componen utilizando una combinación de las formas de expresión mencionadas en la primera y la segunda categorías, mezcladas con hipertextos. Según Di Rosario, en esta categoría de poemas "the reader is presented with a link to click on at the end of every sequence, but these links do not cumulatively build a hypertextual environment, they simply make the reading process advance in a linear way." [Di Rosario, 2011, p. 214]

4. E-Poetry colaborativa que es una nueva forma de poesía que involucra a diferentes personas que no necesariamente se conocen mutuamente. Di Rosario considera dos tipos de colaboraciones. La primera es una

“colaboración cerrada” en que un cierto número de personas (normalmente dos o tres), con destrezas específicas (un poeta, un programador, un artista, etc.) acuerdan colaborar en el mismo proyecto. La segunda es una “colaboración abierta”, en la que los participantes “do not necessarily agree to collaborate on a project, at least not always in advance, they may simply find a webpage where a collaborative poem is instantiated and then decide to collaborate, to leave a mark on that website, with no need to register themselves at the website, to join a group, and perhaps they will contribute with just one text or even with just one letter” [Di Rosario, 2011, p. 292]

5. E-Poetry generativa en la que se construyen poemas diseñando un marco en el que sólo ciertos componentes se llenan al azar con palabras gramaticalmente apropiadas. Di Rosario afirma que “el lenguaje tiene una función pragmática esencial basada en referencias concretas” [Di Rosario, 2011, p. 277]. Además y más allá de las diferentes técnicas (rimas, versos libres, etc.), las diferentes formas (haikús, proverbios, relatos cortos, etc.) y procesos generativos (sintaxis algorítmica, frase matricial, etc.), los diferentes experimentos informáticos se han basado en el mismo principio: la escritura literaria consiste esencialmente en el ordenamiento combinado. Chris Funkhouser remonta la e-poesía generativa a la era antes de WWW: “[a]ssembling texts using different varieties of a slotted framework was pervasive in the prehistoric era of digital poetry”.[Funkhouser, 2007, p. 60]

I-II-I

Prehistoria de la poesía digital

Escribir una historia de la poesía digital en la era de la WWW es muy difícil a causa de la cantidad de las obras publicadas y la falta de acceso a todos los lenguajes donde este género viaja y vive. Escribir una breve prehistoria de la poesía digital parece más viable debido a las limitaciones que la industria impuso sobre el género a través de restricciones de acceso propias de la lógica del consumo de elite. También porque las computadoras no eran tan accesibles como lo son ahora y porque las plataformas de distribución eran extremamente limitadas.

Giovanna di Rosario remonta la historia de la poesía digital al momento de la creación de la escritura. En su perspectiva, la estructura de la escritura se caracteriza por utilizar dos sistemas simultáneamente: el lingüístico y el gráfico. Según ella, la forma en que se codifica esta información lingüística determina una modalidad. La información gráfica se transmite utilizando primordialmente un medio y la lingüística otro. Cada tipo de expresión reside en un dominio independiente: el primero es verbal, el segundo es visual. En su recorrido histórico, menciona la institucionalización de la poesía a través de su práctica desde el Renacimiento y luego destaca las renovaciones radicales ocurridas en el siglo XIX. Di Rosario escribe:

> Normally when talking about visual poetry, we immediately think of Apollinaire's calligram and also of the more recent avant-garde movements. Actually, even if the practice of this type of poetry in the Western tradition is not as rich in breadth and depth as, for instance, the Arabic, the Chinese or Japanese cultures, there are very interesting examples of contamination between the written word and image through the centuries. [Di Rosario, 2011, p. 26]

Di Rosario demuestra que la poesía visual puede ser remontada hasta alrededor de 1700 AC y también que esta tradición nunca fue abandonada por los poetas, incluso después del Renacimiento y de los libros impresos.

Mallarmé y su nueva "sintaxis" del espacio viene después y el futurismo aplica algunos cambios radicales a través de su intención de destruir la sintaxis tradicional. Apollinaire llega a continuación y, como Di Rosario señala, sus caligramas constituyen un puente del pasado al futuro: de la *carmina figurata* a las palabras en libertad. Con Apollinaire la escritura empezó a convertirse en un arte del espacio. El movimiento Dada y el Surrealismo han sido ampliamente analizados en estos aspectos y no hay necesidad de revisarlos al detalle aquí. De esta secuencia de vanguardias, el movimiento Fluxus es uno de los primeros que llega hasta la era de las computadoras. A partir de la década de 1950, Fluxus fue consolidándose como una comunidad internacional de artistas, arquitectos, diseñadores y compositores que produjeron, tanto en el campo de las artes visuales y la música, como en el de la literatura, el urbanismo, la arquitectura y el diseño. Fluxus es considerado como el movimiento artístico más radical y experimental de la década de 1960. Según Maciunas "[es] una manera de hacer las cosas muy informalmente, una clase de

grupo de la broma" [Maciunas, 1998, p. 227]. Fluxus también ha sido considerado como el inicio del Mail-Art.

Di Rosario continúa su explicación con la aparición de internet:

> The arrival of the Internet on the scene has increased accessibility to this trend and brought to it an agility previously unimaginable. What was once a limited group of people, who sent each other ideas and small works of art by post, has grown today to the point where thousands of groups are involved in a broad-based and subversive cultural movement. As an open network, mail-art and e-mail-art offer limitless possibilities. ¨ [Di Rosario, 2011, p. 61]

Por otra parte, Eduardo Kac comienza su cronología de la poesía mediática con un enfoque diferente. El primer evento importante, según Kac, corre a cargo del poeta ruso Velimir Khlebnikov, quien escribe "la radio del futuro" (1921), texto en el que prevé el impacto de las telecomunicaciones sobre la literatura, en particular, y la cultura, en general. El segundo evento que Kac propone es probablemente más conocido, y se refiere a cómo e. e. cummings incorpora el sistema mecánico de su máquina de escribir *Smith-Corona portátil* en la sintaxis visual de muchos de sus poemas. En 1923, cummings publica *Tulips and Chimneys*, su primer libro de poesía.

El evento fundador de la poesía electrónica mencionado por Di Rosario, Kac y Funkhouser, son los textos/poemas de Théo Lutz, generados a través de una computadora en 1959: *Stochastische texte*. Todos también reconocen el lugar de OULIPO, fundado por Raymond Queneau y François Le Lionnais (Ouvroir de Littérature Potentielle, o taller de literatura potencial) en Francia, que viene después del trabajo de Théo Lutz, en 1960. Como ya mencioné, OULIPO fue un grupo de

escritores y matemáticos que incluyó como miembros a Claude Berge, Georges Perec, e Italo Calvino, entre otros.

Funkhouser narra la historia de la poesía digital del año 1959 con más profundidad y detalle que Di Rosario y Kac. Al hablar sobre poesía generativa, se concentra en los poetas de movimiento Dada y menciona el manifiesto de Tristan Tzara "Cómo escribir un poema Dada", donde el autor da instrucciones a sus lectores para que corten artículos de periodísticos en palabras individuales y con ellas hagan un poema seleccionando y reorganizando estas palabras al azar. Según Funkhouser:

> Dadaists challenged convention with other methods, including collage, the invention of new words (neologism), typo-graphical distortion and desecration, transcription (or use) of nonsemantic sounds, and collaboration. Since all of these elements are found in early digital poems (i.e., the language and other media that serve as the data-base of the generated digital poem are akin to the clipped words from the newspaper, etc.), Dadaism is unquestionably a historical model that can be used as context for many of the computerized works. [Funkhouser, 2007, p. 33]

Funkhouser también cita la poética digital de Glazier, en la que este último subraya la importancia de la relación de OULIPO con la poesía digital y su poética:

> Oulipian invention provides a rigorous investigation of the program as a generative agent in the literary work, and its methods provide a useful reference point for considering algorithmic generation of poetry. [Funkhouser, 2007, p. 34]

Funkhouser no se centra en la historia de las expresiones visuales en la poesía, como lo hace Di Rosario, y comienza su relato a partir de la influencia de Mallarmé para luego continuar con la mención de otras exploraciones de semiótica visual realizadas por la poesía concreta en Alemania y Brasil, en la década de 1950. Según

Funkhouser, “The work of e. e. cummings also presumably helped to liberate lines and formations of poetry from strict arrangement.” Funkhouser continúa:

> Olson’s concept of “composition by field,” introduced in the essay “Projective Verse” (1950), which argues against inheriting the “line, stanza, over-all form” of “old” poems and emphasizes “kinetics” (“a high energy-construct and, at all points, an energy discharge”), “principle” (“right form, in any given poem, is the only and exclusively possible extension of content under hand”), and “process” (“how the principle can be made so to shape the energies that the form is accomplished”) also indicates that poets sought to break away from tradition by establishing a different look for literary forms. Projective works, in Olson’s view, use the “machine as a scoring to his composing, as a script to its vocalization”. Olson’s perspectives on the use of the typewriter, which “sound a call for the scriptural imagination to engage the materiality under one’s fingers,” writes Glazier, suggest that “literary form can be revitalized.” [Funkhouser, 2007, p. 88]

Otra categoría de E-Poetry que Funkhouser describe es la poesía del hipertexto. Aunque Michael Joyce afirma que “hypertexts can never be adequately represented in print” [Joyce, 1996, p. 21], varios libros se refieren a menudo a diversos autores como precursores del hipertexto. Entre los primeros textos impresos citados por Nelson como hipertextuales están *Tristram Shandy* de Laurence Sterne, *Rayuela* de Julio Cortázar, y *Spoon River Anthology* de Edgar Lee Masters. El hipertexto como estructura se puede rastrear incluso en la antigüedad. *Las mil y una noches* contiene tal estructura: cada cuento contiene algunos personajes y cada personaje del cuento narra un nuevo cuento que contiene nuevos personajes que tienen sus propios cuentos. En la siguiente sección hablaré de tres teorías retóricas relacionadas con las categorías antes mencionadas de la poesía electrónica.

I-II-II

La retórica digital: tres teorías principales

El lenguaje de los nuevos medios

Lev Manovich comienza el primer capítulo de su libro *The Language of New Media*, con una simple pregunta: "¿Qué es New-Media?" Su discusión implica la comprensión popular de este concepto desde los nuevos medios hasta los mitos comunes que circundan los discursos del nuevo periodismo de los medios de comunicación. Muchos ejemplos que utiliza el autor se basan en el desarrollo del cine, cuyo enorme impacto cultural en nuestra sociedad nadie adivinó en los primeros días de su invención. Manovich enlista cinco principios para los nuevos medios:

1. Representación numérica. Todos los objetos nuevos de los medios de comunicación están compuestos por códigos digitales. Este hecho tiene dos principales consecuencias:

a. Un nuevo objeto de los medios puede ser descrito formalmente (matemáticamente).

b. Un nuevo objeto de los medios de comunicación está sujeto a manipulación algorítmica; es decir, los medios de comunicación se vuelven programables.

Por ejemplo, los primeros dígitos del objeto ejecutable –es decir, el archivo exe- de *Patchwork Girl* son estos: 1001101101101010010000. Los archivos

exe son códigos de máquina binarios que se han compilado a partir del código fuente por un programa que se ejecuta en modo usuario pero produce un archivo que es compatible con el sistema operativo para ejecutarlo en modo kernel, en el que el hardware reacciona con la producción de textos y imágenes en la pantalla.

2. Modularidad. Se define como “fractal structure of new media where a new media object has the same modular structure throughout.” Por ejemplo, Manovich menciona a la programación orientada a objetos cuando dice: “Many new media objects are in fact computer programs that follow structural programming style” [Manovich, 2001, p. 52]. En el código fuente de Gnoetry, un tanka y un haiku son formas poéticas que se definen con la misma modularidad y con un objeto de *media* llamado *class*:

```
class PoemChoice_Haiku(PoemChoice):
   def __init__(self):
      PoemChoice.__init__(self, "Haiku")
   def get_description(self):
      return "5-7-5"
   def build_poem(self):
      return gnoetics.Haiku()


class PoemChoice_Tanka(PoemChoice):
   def __init__(self):
      PoemChoice.__init__(self, "Tanka")
   def get_description(self):
      return "5-7-5\n7-7"

   def build_poem(self):
      return gnoetics.Tanka()
```

3. Automatización. Los dos primeros principios permiten la automatización de muchas operaciones para la creación, manipulación y acceso de los medios de comunicación.

4. Variabilidad. Un objeto nuevo de multimedia no es algo fijo, sino algo que puede existir en versiones diferentes, potencialmente infinitas. La variabilidad trae consigo algunos casos particulares que Manovich categoriza como sigue:

a. Los elementos multimedia se almacenan en una base de datos de los medios de comunicación. Manovich considera A las “bases de datos” como una forma cultural reciente.

b. La separación de los niveles de “contenido” e “interfaz” ha llegado a ser posible, es decir, un número de diferentes interfaces puede crearse desde los mismos datos.

c. La información sobre el usuario puede utilizarse para personalizar automáticamente la composición de los medios de comunicación y la creación de sus elementos.

d. Menú basado en la interactividad. El menú de un programa puede cambiar de acuerdo con el análisis de las necesidades de los usuarios y de acuerdo con sus interacciones con el programa. No hay necesariamente un menú fijo en todo el programa.

e. Hipermedia, una nueva estructura de los medios de comunicación, es una extensión del hipertexto en la que el texto se extiende a imágenes, videos, sonidos e interacciones.

f. Actualizaciones periódicas. Un programa siempre es algo inacabado. Se enfrenta a errores o cambios en los sistemas operativos o el hardware, o a las necesidades de los usuarios. Por lo tanto, una nueva versión del programa aparece como una actualización cuando es necesario.

g. Escalabilidad. Las diferentes versiones de los mismos objetos de medios pueden ser generadas en varios tamaños o niveles de detalle. Se puede desarrollar una versión para resolver los errores y problemas existentes que tiene el programa o para satisfacer las nuevas necesidades del usuario. Por lo tanto, una nueva versión puede introducir conceptos completamente nuevos para el usuario o proporcionar un mejor rendimiento de la versión existente.

5. Trans-codificación. Los estudios de los medios de comunicación son más bien “estudios del software”. Un objeto digital puede ser codificado en diferentes formas. Por ejemplo, una canción puede tener el formato de mp3 o wav. Cada uno de ellos utiliza una codificación diferente y están diseñados para diferentes propósitos. Mp3 es para usuarios no profesionales que no necesitan toda la información relacionada con una canción y, por lo tanto, es una versión compacta de la misma canción. El formato wav almacena toda la información (como detalles sobre las frecuencias) para un uso profesional.

Al enumerar estos principios, Manovich abre una discusión sobre qué no son los nuevos medios de comunicación, desafiando los mitos comunes sobre las características de los nuevos medios, como la representación digital *versus* la

representación analógica; o el surgimiento de nuevas formas de comunicación, como las presentaciones multimedia; o las posibilidades de pérdida de información en la digitalización; o, incluso, la mera interactividad.

Manovich utiliza el cine y el lenguaje cinematográfico para abordar estos mitos y mostrar que tales características existían en el cine, porque el cine, por ejemplo, desde un principio se basó en muestras. Cada muestra es una unidad de edición de la narrativa que una película cuenta. El cine nos da la posibilidad de revisar, editar y cambiar los hechos y, por lo tanto, crea un espacio para imaginar diferentes posibilidades de interpretaciones, cada una basada en una edición de las unidades elegidas como una muestra. Sin embargo, los medios digitales amplían esta posibilidad a través de la interacción.

Manovich comienza su investigación sobre el lenguaje de los nuevos medios con una interfaz cultural de los nuevos medios llamados HCI (Human-Computer Interface), cuyo lenguaje propone como compuesto principalmente por elementos de otros lenguajes, que ahora ya son formas culturales familiares, y discute sobre el impacto de la “interfaz de la página” y luego la del “cine”. En el desarrollo de tal nueva interfaz cultural, considera tres formas culturales mayores como las más importantes para describir el lenguaje de los nuevos medios: impresión, cine y la interfaz persona-computadora. Para Manovich, el lenguaje de interfaces culturales es híbrido; sin embargo, afirma que debido a la pantalla, la entrada al mundo virtual es fija, los seres humanos se vuelven inmóviles: “A screen is still a screen... As was the

case centuries ago, we are still looking at a flat, rectangular surface." [Manovich, 2001, p. 31]

La teoría de hipertexto

George P. Landow, en *HyperText: The Convergence of Contemporary Critical Theory and Technology*, como lo sugiere el título, trata de mostrar la convergencia de la teoría crítica –especialmente en términos de las propuestas de Derrida, Barthes y Foucault– y la tecnología –conforme a Nelson, Bush y Van Dam– centrándose en diferentes aspectos de tres agentes principales: "texto", "lector" y "escritor".

Landow muestra diferentes aspectos de tal convergencia describiendo el cambio de terminología en la teoría crítica y el uso de "enlace", "web", "red", "entretejido" y "assemblance", como lo describe Derrida: "We abandon conceptual systems founded upon ideas of center, margin, hierarchy, and linearity and replace them with ones of multilinearity, nodes, links, and networks". [Landow, 2006, p. 1]

En un aspecto conceptual, si Barthes, por un lado, pone de relieve el texto legible y su no-linealidad, desdibujando las fronteras de "lector" y "escritor", por el otro, Derrida subraya la apertura textual, la intertextualidad y la irrelevancia de la distinción entre el dentro y el fuera de un «texto». El libro trata de mostrar cómo la multivocalidad, según Bajtín, y la descentralizaron, según Derrida, se pueden aplicar a un fenómeno denominado "Hipertexto".

Desde este punto de vista, el “Hipertexto” es hijo de tres reconfiguraciones entretejidas: las del texto, el lector y el escritor. Y tales reconfiguraciones tienen su política. Estas reconfiguraciones marcan el proceso de transición de “texto” a “hipertexto”.

En primer lugar, el “hipertexto” implica un lector más activo, un lector que tiene la oportunidad de leer como un autor. Elementos visuales como el cursor amplían la cantidad y diversidad de información no-verbal en el texto e implican la presencia del lector en el mismo. El «Texto» es disperso y atomizado. Los enlaces proporcionan diferentes vías a través de un cuerpo dado de *lexias* o fragmentos de texto. La fragmentación aumenta y la idea de “el texto” como unidad textual se difumina y deja de ser válida. En contraste con los textos impresos, el hipertexto ofrece al menos dos tipos diferentes de inicio: el primero se refiere a la *lexia* individual y el segundo a una reunión de *lexias* en un meta-texto. El hipertexto incluso desdibuja los límites extremos del meta-texto y se mantiene con una composición abierta, ampliable e incompleta. Se redefine el centro, al negarse a otorgar centralidad a cualquier cosa, a cualquier *lexia*. La centralidad, al igual que cualquier otro tipo de valores, como la belleza y la importancia, reside en la mente del espectador, y una disolución de la centralidad vuelve al medio potencialmente democrático, transforma el medio en “a model of society of conversations in which no one conversation, or no one discipline or ideology, dominates or founds the others” [Landow, 2006, p. 123]. Landow afirma el impacto de dicha reconfiguración de los textos en las dimensiones sociopolíticas:

> The basic experience of text, information, and control, which moves the boundary of power away from the author in the direction of the reader, models such a postmodern, anti-hierarchical medium of information, text, philosophy and society. [Landow, 2006, p. 3]

Literatura ergódica

En una entrevista, Espen J. Aarseth habla de sus intenciones en la construcción de dos palabras clave principales del libro *Cybertext: Perspectives on Ergodic Literature*. Afirma:

> In constructing the term Cybertext, I wanted to show that there are many forms and machinations of text, of which hypertext (in the node-link sense) is just one. Hypertext was at that time (early 90s) getting all the attention, and I wanted to change that, by constructing a larger perspective. But both "hypertext" and "cybertext" are, it seems to me, ideological constructs rather than actual technologies. The term Ergodic, on the other hand, is an attempt to define the quality that is so inapropriately usually referred to by "interactive"; a hopelessly unfocussed and ill-conceived term, completely void of analytic meaning. [Aarset, 1999]

Aarseth toma prestado el término “ergodicidad” de la física, aunque la palabra se deriva, en última instancia, de dos términos griegos, ergon (trabajo) y odos (ruta) porque trata de describir cualquier texto de acuerdo con su modo de recorrido por variables como “dinámica”, “determinabilidad”, “perspectiva de transitoriedad”, “acceso”, “vinculación” y “funciones de usuario”. Sin embargo, cibertexto, un término que Aarseth considera como una construcción ideológica, tiene la intención de ser un punto de vista sobre todas las formas de textualidad incluyendo la hipertextualidad, un modelo de comunicación textual que se acomoda a cualquier

tipo de texto. Aarseth elige cuatro tipos de cibertexto para trabajar: hipertextos, juegos de aventura en equipo, textos generados por computadora, y entornos colaborativos llamados MUD. Por eso, sustituye “el texto” con las “lecturas” y ofrece una máquina textual que involucra a los signos verbales, al medio y al operador. En este enfoque, el texto no es una cadena de significantes, como para los lingüistas y semiólogos. El prefijo “cibernético”, indica que el texto se ve como máquina. Cibertexto nos lleva a un cambio de paradigma del trío autor / remitente, texto / mensaje y lector / receptor, a la conversación cibernética entre los diversos agentes que participan en la máquina textual.

La clasificación y la observación de las expresiones de la superficie de los signos cibernéticos parecen insuficientes para la comprensión de un cibertexto. Hay que considerar dos niveles interconectados de código y de expresión. A partir de estas definiciones se puede comenzar a cuestionar la mayor parte de las ideas establecidas de la ideología del hipertexto, como la no linealidad y la interactividad. Aaresth afirma que dentro de la ideología del hipertexto, la palabra “Interactividad” opera textualmente en lugar de analíticamente. Al aceptar la dicotomía de Eco, entre las obras "abiertas" y las "cerradas", abordando especialmente la idea de "obras en movimiento", porque consisten en unidades estructurales no planificadas o físicamente incompletas, en su libro *Cybertext: Perspectives on Ergodic Literatura*, Aarseth adopta la idea de cyborg de Donna Haraway en el desarrollo de una estética Cyborg. Aarseth considera al cyborg como una figura nacida de la interferencia entre el autómata y la autonomía. Esa estética desafía el viejo dualismo occidental de yo /

otro, alma / cuerpo y así sucesivamente. El texto como máquina implica una estética equivalente. Una estética cyborg no planea crear obras maestras de la literatura. Aarseth cree que el narcisismo es un elemento necesario en el proceso artístico, que es un proceso autorreflexivo y autocrítico. Una máquina no puede criticar o valorar su propio trabajo; por lo tanto, la máquina en sí nunca será un buen autor tradicional. Por eso, una poética del cibertexto no considera el género literario tradicional ni los formatos como ideales de la nueva literatura, y no implicará un uso acrítico de la teoría literaria tradicional, en la crítica de la literatura participativa.

II

Ontología de la poesía digital

En la introducción y también en el primer capítulo de este estudio he discutido varios factores subyacentes en la distancia que separa la poesía digital del discurso tradicional de la poesía en los estudios literarios. Markku Eskelinen también confirma que la falsa dicotomía entre la cultura impresa y los medios digitales sigue dividiendo el campo académico de los estudios literarios. Incluso los eruditos literarios más prominentes evitan textos digitales y ergódicos y permanecen con los trabajos impresos y no-ergódicos. Dice Eskelinen:

> On the other side of that divide, scholars of digital literature tend to focus only on digital specimens, even though a limited selection of print literature is usually mentioned and included in the discussions, most likely as predecessors to main object of study providing a tradition and all other benefits that come with that territory. [Eskelinen, 2012, p. 3]

Según él, las principales razones por las que la división sigue presente pueden ser discutidas considerando los siguientes factores:

1. El negocio de la academia, donde los eruditos tienen que especializarse y prefieren permanecer dentro de sus áreas primarias de especialidad.

2. La pandilla de la impresión, que por razones históricas tiene prerrogativas, culturales, económicas, institucionales, teóricas y así sucesivamente, y que todavía cree que la literatura impresa es la única literatura que tiene valor. La literatura digital contiene aspectos que se oponen a la gama de supuestos básicos de una amplia variedad de teorías sofisticadas de la literatura impresa. Muchas de las teorías generales de la literatura impresa son o pueden ser válidas solamente en el contexto de la literatura impresa y, por lo tanto, los eruditos de la impresión tienen miedo a ser desafiados.

3. Los eruditos orientados a la literatura ergódica y digital no han desafiado a los eruditos de la literatura impresa con análisis y perspectivas derivadas de lo digital y de otras "anormalidades" y por lo tanto los paradigmas basados en la impresión aún permanecen en el poder.

Eskelinen también menciona cuatro etapas traslapadas en las discusiones teóricas alrededor de los estudios literarios de los medios:

1. Las poéticas de teóricos que trabajan en el campo de los generadores de texto (por ejemplo Max Bense), poesía digital (por ejemplo Pequeno Glazier), intermedia (por ejemplo Dick Higgins), ficción del hipertexto (por ejemplo Michael Joyce), holopoesía (por ejemplo Eduardo Kac), videopoesía (por ejemplo Melo e Castro) y ficción interactiva (por ejemplo Montfort). Eskelinen los ve como teóricos que se contentaron con explorar los

potenciales de un medio particular, género o tecnología material, sin generalizar sus descubrimientos a otras áreas y sin ninguna intención de construir una teoría o perspectiva comparativa.

2. Teóricos cuyo enfoque está en la división de lo análogo (impreso o antiguo) y de lo digital (o nuevo) y podrían enumerar varias propiedades clave de los nuevos medios (por ejemplo Manovich o Murray).

3. Teóricos que trataron de producir una introducción a una teoría comparativa de los medios de comunicación, dentro de la cual se podía situar cualquier texto literario. Estos teóricos tratan de desplazar el énfasis lejos del esencialismo mediático (lo que un medio de comunicación teóricamente *es*) hacia lo que un medio de comunicación *hace* (por ejemplo Aarseth).

4. Enfoques que tratan de ir más allá de la superficie textual y los modelos comunicativos en general hacia las operaciones, lógicas operacionales y procesos de varios medios digitales (por ejemplo, Boost, Bootz y Wardrip-Fruin).

Según Eskelinen, “todos estos enfoques no han sido plenamente desarrollados y su poder explicativo y su valor heurístico aún no están claros.” [Eskelinen, 2012, p. 4]. No es sólo el discurso general de los estudios literarios el que refleja las divisiones mencionadas entre lo impreso y lo digital. La pedagogía de la escritura creativa y otros enfoques pedagógicos también reflejan la misma división. La pedagogía de la escritura creativa ignoró sobre todo la literatura digital. Entre las teorías de la

escritura, el modelo de proceso de Donald M. Murray fue un acercamiento muy común a la teoría de la escritura en los años 80, cuando la escritura se consideraba como resultado de un proceso de preescritura, redacción, revisión y edición. Dentro del campo de la informática, los lenguajes procesuales compartieron el mismo destino, teniendo como modelo el proceso, y perdieron la popularidad a la luz de los críticos emergentes de las teorías postprocesuales y postmodernas, durante los años 90, muy similares al paradigma de programación estructurada, después del crecimiento de la programación orientada a objetos. Al respecto, John H. Whicker argumenta que:

> At the turn of the century, writing theory, those theories attempting to explicitly theorize writing itself, seems divided among 1) attempts to reconcile post-process with process or epistemic rhetoric in a continuance of a social constructivist Process Paradigm, usually with a committed emphasis on pedagogy, 2) extensions of social process models based in activity theory largely focused on genre and its role in mediated literate action, and 3) a convergence of more radical theories through an oppositional critique of both liberatory pedagogy and postmodern cynicism, and attempts to overcome this binary opposition through critiques and re-articulations of agency and subjectivity that often draw on radical postmodern and poststructural theories. At the same time, this last current and its attempts to confront issues of subjectivity and agency in writing and rhetoric also includes the emergence of theoretical efforts to reject the Process Paradigm through post-human and ecological and complexity theories. [Whicker, 2014, pp. 69-70]

Sin embargo, esta oposición binaria -la dualidad “proceso versus producto”-, todavía juega un papel importante en el análisis de la poesía digital, como una forma de escritura. Una forma cultural como la poesía digital no se puede leer sólo como un producto (un programa y sus resultados) sin considerar el proceso de desarrollo del

mismo. Al igual que muchas expresiones artísticas contemporáneas, como el arte conceptual, el foco de la atención se desvía del producto hacia el proceso.

Este capítulo y el siguiente tratan de proporcionar un marco teórico para la lectura de la poesía digital, teniendo en cuenta tanto el proceso como el producto. Para ello, usaré un acercamiento ontológico al proceso de escritura de la poesía digital. Es decir, intentaré sugerir una ontología de escribir poemas digitales (proceso) paralela a una ontología de lectura de poemas digitales (producto). Además, para evitar otra oposición binaria -la dualidad profundidad vs superficie-, voy a discutir el código fuente de una serie de poemas digitales para demostrar las dimensiones de las interacciones entre un poema y su código fuente. Por lo anterior, voy a discutir primero los aspectos ontológicos de un poema digital y los discursos creativos tras él.

Según Thomas Gruber, “una ontología es una especificación explícita de una conceptualización”. [Gruber, 1993, p. 199] Gruber afirma que, en filosofía, el término “ontología” significa un recuento sistemático de la existencia. Si esto significa una clasificación de entidades, entonces esta caracterización áspera de la ontología filosófica es aceptable, aunque para un público interdisciplinario requiere un mejor esclarecimiento.

Barry Smith y Christopher Welty explican que la ontología filosófica es la ciencia de lo que es, de los tipos y estructuras de objetos, propiedades, eventos, procesos y relaciones en cada área de la realidad. Según ellos, la ontología filosófica toma muchas formas, desde la metafísica de Aristóteles hasta la teoría de objetos de

Alexius Meinong. El término “ontología” fue acuñado en 1613, de manera independiente, por dos filósofos, Rudolf Göckel (Goclenius), en su *Lexicon Philosophicum*, y Jacob Lorhard (Lorhardus), en su *Theatrum Philosophicum*. El *Diccionario* de 1721 de Bailey, define la ontología como “la investigación del ser en tanto que es en su abstracción”.

Sin embargo, el surgimiento de la ontología en la informática, como lo explican Smith y Welty, refleja una victoria del contenido sobre el proceso. Es decir:

> A victory which has been, somewhat paradoxically, reinforced as a result of the fact that, as software itself has become ever more sophisticated, software developers and computer theorists have increasingly found it possible to focus on the data upon which their systems operate rather than on the functionality and procedural aspects of the systems themselves. [Welty, 2011, p. 4]

Lo que Smith y Welty llaman “victoria del contenido sobre el proceso”, yo lo llamo “hegemonía del producto sobre proceso”.

Un acercamiento ontológico también tiene el peligro de caer en un esencialismo, como Eskelinen ha señalado antes, por lo tanto intento tomar algunos contrapuntos para no caer en el esencialismo de los medios de comunicación y también en la dualidad de producto/proceso. Los contrapuntos que estoy aplicando para este propósito están arraigados en el enfoque que he tomado acerca de la creatividad, el enfoque de Mihaly Csikszentmihalyi, que adopta explícitamente una visión de sistemas. Mientras que muchos estudios de creatividad comienzan haciéndose la pregunta acerca de qué es la creatividad, Csikszentmihalyi se pregunta dónde está la creatividad.

Antes de eso, proporcionaré algunas explicaciones más sobre la ontología y el desarrollo de la ontología.

II-I

Sobre el desarrollo de una ontología informática

Una ontología informática es una especificación explícita de una conceptualización. El término es un préstamo de la filosofía, donde una ontología es una investigación sistemática de la existencia y trata de las categorías básicas del ser. La palabra griega original de la cual deriva la palabra "ontología" significa estudio o conocimiento de lo que es o existe, y la rama filosófica con el mismo nombre ha tomado el término como referencia a la realidad que nos rodea, independientemente de nuestra propia visión sobre ella. En otras palabras, la ontología requiere que hagamos una distinción estricta entre el sujeto observador y el objeto observado. En informática, una ontología es una especificación formal y explícita de un vocabulario o conceptualización compartida, que se puede usar para modelar un dominio del conocimiento.

En su uso moderno, la ontología ha conservado su significado original, pero también tiene un objetivo práctico definido. Sirve para proporcionar una base para la comprensión común de algún área de interés entre una comunidad de personas que pueden no conocerse entre sí y que pueden tener antecedentes culturales muy diferentes. Por ejemplo, para los sistemas de Inteligencia Artificial (IA), lo que “existe” es lo que puede ser representado. Cuando el conocimiento de un dominio está representado en un formalismo declarativo, el conjunto de objetos que pueden ser representados se llama “universo del discurso”. Este conjunto de objetos, y las

relaciones descriptibles entre ellos, se reflejan en el vocabulario representativo con el que un programa basado en el conocimiento (un programa experto ideado para crear nuevo conocimiento a partir del conocimiento explícito presente en los documentos de las bases de datos) representa el conocimiento. Así, en el contexto de la IA, podemos describir la ontología de un programa definiendo un conjunto de términos de representación. En tal ontología, las definiciones asocian los nombres de las entidades en el universo del discurso (por ejemplo, clases, relaciones, funciones u otros objetos) con un texto legible para el hombre, que describe los significados de los nombres, y también los axiomas formales que restringen la interpretación y el uso bien formado de estos términos. [Gruber, 1993, pp. 201-203]

Según el estándar W3C - World Wide Web Consortium -, la web semántica es una visión para el futuro de la web, en la que se da un significado explícito a la información, facilitando a las máquinas el procesamiento e integración automáticos de la información disponible en la Web.

Wikipedia y el nacimiento de los sistemas semánticos abrieron un nuevo campo de investigación que involucra tanto los sistemas de desarrollo de la ontología como las redes sociales. Los wikis, como sistemas de gestión de contenido colaborativo, llegaron a ser populares por el éxito del proyecto de Wikipedia y fueron utilizados, por ejemplo, como gestión del conocimiento o sitios web de la comunidad. La adición de un modelo subyacente de conocimientos descritos en las páginas wiki fusiona la administración de contenido colaborativo con el poder de la web semántica. Después de 2005, cuando las wikis semánticas se implementaron

seriamente, muchos de estos sistemas están siendo propuestos, modelados, investigados y usados.

No hace mucho, pero antes de la era digital y la invención de los motores de búsqueda, cualquier estudio comparativo sobre el uso de una frase simple en las obras de Rumi o Cervantes, era verdaderamente un dispendio de tiempo. Ahora, una simple búsqueda en sus obras nos puede dar suficiente información para este tipo de estudio. Hablando de estos estudios, nos enfrentamos a la cuestión de la recuperación de información o RI. En el año 2001, cuando Berners-Lee, Hendler y Lissila estaban escribiendo acerca de un sueño, el sueño de la web semántica, tal vez no alcanzaban a imaginar cuántas personas, en diferentes idiomas, compartían el mismo sueño. Berners-Lee Escribe:

> For the Semantic Web to function, computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning. [Berners-Lee, 2001].

La “interoperabilidad”, o la capacidad de los diferentes sistemas tecnológicos para intercambiar los datos, es el núcleo de tal sueño. Para ello, dos sistemas deben compartir el mismo vocabulario para poder comunicarse.

Según el Consorcio World web (W3C):

> The Semantic Web is about two things. It is about common formats for integration and combination of data drawn from diverse sources, where on the original Web mainly concentrated on the interchange of documents. It is also about language for recording how the data relates to real world objects. That allows a person, or a machine, to start off in one database, and then move through an unending set of databases which are connected not by wires but by being about the same thing. [W3C, 2013]

La web semántica utiliza ontologías para modelar y representar información para que las máquinas y los seres humanos puedan utilizarlos de manera cooperativa. Usualmente, se utiliza una estructura gráfica para representar las ontologías. El gráfico consiste en:

1. Un conjunto de conceptos (vértices en un gráfico),
2. Un conjunto de relaciones que conectan conceptos (bordes dirigidos en un gráfico), y
3. Un conjunto de instancias asignadas a un concepto particular (registros de datos asignados a conceptos o relaciones).

Aunque dentro de la filosofía las ontologías se utilizan para describir el mundo, clasificarlo y categorizarlo, en el procesamiento del lenguaje natural, el objetivo de estas otras ontologías es modelar el conocimiento léxico y de dominio. En la web semántica, deben proporcionar modelos de interpretación para los recursos de la web.

Existen varios lenguajes de ontología para la codificación y la representación de ontologías. OWL o Web Ontology Language es uno de estos lenguajes. Una ontología de OWL consiste en una secuencia de anotaciones, de axiomas, y de hechos. Una ontología puede tener un nombre y las anotaciones son utilizadas para registrar la autoría y otra información asociada, y también referencias a otras ontologías. Los hechos contienen cualquier información sobre un individuo en particular, en forma de clases a que el individuo pertenece y más propiedades y

valores de ese individuo. Los hechos también se utilizan para que los identificadores individuales sean iguales o distintos. Los axiomas se utilizan para proporcionar información sobre clases y propiedades.

También son diversas las herramientas disponibles para el desarrollo de ontologías. Una de ellas es Protégé, una IDE (Integrated Development Environment) de ontología desarrollada por la Universidad de Stanford. Generalmente la IDE proporciona un editor para código fuente, herramientas de automatización para compilar y ejecutar el código fuente, y un programa depurador. Protégé ofrece unainterfaz gráfica interactiva.

II-II

Bosquejos de una ontología de un poema

Al igual que cualquier otra ontología, una ontología para escribir un poema digital debe ser completa (sin falta de cobertura), inequívoca, intuitiva (tomada de un dominio apropiado), genérica (permitir diferentes usos en diferentes contextos) y extensible. Teniendo en cuenta el dominio, el poema digital, la ontología podría iniciar de las siguientes declaraciones:

A. Un poema digital es un poema.

Un poema digital es un artefacto que tiene una materialidad diferente a lo que se denominaba poema durante la cultura impresa u oral. Aunque parezca una declaración trivial, no lo es. Porque un lado de la expresión se refiere a un nuevo artefacto y el otro lado se refiere a un concepto existente e histórico.

B. Un poema digital contiene software.

No dije que “un poema digital es un software” sino que lo contiene para poder considerar casos de poesía digital performativa o instalaciones que utilizan un software diseñado para un espacio específico o un lector/jugador específico que interactúa o juega con una máquina para generar poesía. En tales casos performativos, el poema está incompleto sin el espacio específico o el actor específico. Hay varios ejemplos de mezclas de poesía digital con danza u otras artes

escénicas. La experiencia de la poesía en tales obras no se puede reducir sólo al programa o la actuación. Al mismo tiempo, este enfoque me permite deconstruir dualidades antes mencionadas: ¿Dónde yace la superficie de un poema digital y dónde comienza su profundidad? o ¿cuál es la esencia y la apariencia de un poema digital? ¿La máquina se puede separar del texto o no?

Estoy consciente de que cualquier definición de la poesía se enfrenta a la ambigüedad y a la incompletitud. Los diccionarios están llenos de tales definiciones. Collins explica que "a poem is a piece of writing in which the words are chosen for their beauty and sound and are carefully arranged, often in short lines which rhyme", o Cambridge define un poema así: "a piece of writing in which the words are arranged in separate lines, often ending in rhyme, and are chosen for their sound and for the images and ideas they suggest." S. R. Levin afirma que "Poetry is a literary form in which language is used in a concentrated blend of sound and imagery to create an emotional response." [Hertz, 1997, p. 8]

Similar a Levin, Wardrip-Fruin habla de las artes que llaman nuestra atención sobre el lenguaje, nos presentan personajes, nos cuentan historias, y nos hacen reflexionar sobre las estructuras y prácticas comunes de tales actividades. La ambigüedad y la incompletitud son partes de estas definiciones positivas. Por ejemplo, en tanto la belleza es subjetiva, algunas palabras son hermosas para algunos lectores y algunas no lo son, y por lo tanto un pedazo de escritura puede ser considerado poema para un grupo y para otro no. La rima, el sonido, la imagen y la idea existen simultáneamente en otros textos, como en la filosofía o en el teatro y, sin

embargo, no son considerados poemas. Cualquier otro texto con una atención concentrada en el lenguaje también puede producir una respuesta emocional. Las definiciones negativas -un acercamiento muy común entre poetas- se originan ante la desesperación de la existencia de muchísimas definiciones positivas.

La historia de la literatura también nos muestra una gran variedad de textos considerados como poemas. Parece que la única definición genérica de un poema es cuando un lector llama a un texto *poema*. Aquí se levantan otras discusiones sobre el asunto, tal como la autoridad y las dimensiones filosóficas de la palabra "llamar" como "nombrar, hacer presente o invocar". Sin embargo, todas las ambigüedades mencionadas están ausentes cuando "una persona llama a un texto, *poema*", por lo menos para ella. La autoridad de llamar a un texto *poema* no está siempre en el escritor. Por ejemplo, Ahmad Shamlou, el principal poeta iraní del siglo pasado, lanza un texto al bote de la basura y su colega crítico lee accidentalmente el texto, lo llama *poema* y convence al poeta de que lo incluya en su libro. "Convencer a alguien de que un texto es un poema", al mismo tiempo, abre nuevos debates sobre las cuestiones de las subjetividades y las autoridades, que no son objeto de mi estudio en este capítulo. El acercamiento de Peter Gendolla a la literalidad también cae en las mismas ambigüedades. Gendolla escribe:

> All of these fictitious realizations generate something that is nonexistent. They generate a paradoxical, self-destructive and self-constructive figure. This is what I mean by literariness: In literature there originates a unique, purely aesthetic distinction compared with all historical, normative, or functional meanings and uses of language. [Ricardo, 2009, p. 169]

¿Quién tiene la autoridad para reclamar la singularidad de un texto? Es un lector como Peter Gendolla quien tiene la autoridad de percibir la singularidad del texto. Sin embargo, el concepto de singularidad de Gendolla puede ser referido a la "creatividad". Un texto debe ser considerado creativo para ser un poema.

Por lo tanto, añado otra declaración en mi acercamiento a esta ontología:

C. Un poema es un objeto que por lo menos un lector llama "poema".

Hablar de objetos nos permite tomar un camino más extensible y genérico hacia el desarrollo de una ontología. También por esto agrego una ontología de la lectura a una ontología de la escritura cuando se trata de poesía. Es decir, la lectura es inseparable de la creación y existencia misma de un poema.

Katherine Hayles afirma que en la poesía digital "the medium lends itself to experimental practice, especially to forms that disrupt traditional notions of stable subjectivities and ego-centred discourses" [Hayles, 2008, p. 17]. Por lo tanto, un acercamiento a la creatividad que sólo se concentra en los individuos o en la creatividad centrada en el ego y evita las creatividades interactivas y colectivas no puede ser adoptada en este discurso. Utilizo el modelo de Csikszentmihalyi, donde se afirma que la creatividad resulta de la interacción de un sistema compuesto por tres elementos: una cultura que contiene reglas simbólicas, una persona que aporta novedad al dominio simbólico y un campo de expertos que reconocen y valúan la innovación. "Innovación" o "novedad" se considerará como "unicidad" en la

definición de literalidad de Gendolla. Paralelamente a esto, la autoridad en la cual un objeto se llama “poema” se origina de uno de estos tres agentes: un individuo, un dominio o una cultura. Vuelvo a escribir la declaración c para adjuntarle la siguiente declaración:

C. Un poema es un objeto creativo que por lo menos un lector llama “poema”.

D. Un lector es un individuo, un dominio o una cultura.

Por otro lado, un objeto creativo es el resultado de un proceso creativo. El individuo creativo detrás de un poema puede ser una persona, un equipo o una máquina o un conjunto de todos ellos. Además, para no separar el producto del proceso, incluyo ambos procesos de lectura y escritura dentro del objeto del poema. Así se genera la siguiente declaración:

E. Un poema contiene un proceso de lectura y un proceso de escritura.

II-III

El objeto y el proceso creativo

La idea central de la ontología, aparte del proceso de lectura y escritura y la objetividad del poema, reside en la novedad, la singularidad y la creatividad de la escritura. En otras palabras, el elemento crítico indispensable para calificar a algún producto de “creativo” es que sea nuevo.

Para entender los tres componentes de Csikszentmihalyi, la figura de abajo puede sernos útil.

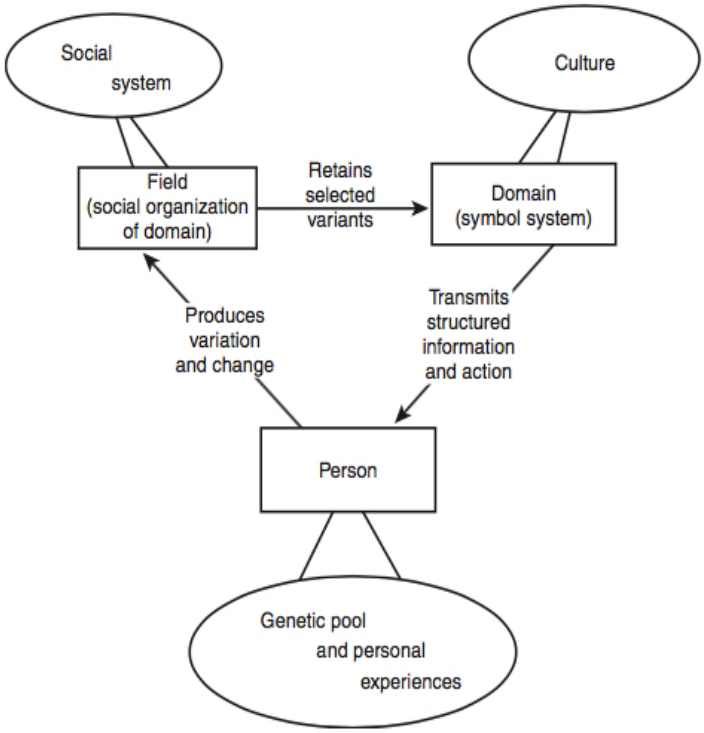

Fig. 17: los componentes de creatividad según Csikszentmihalyi [Weisberg, 2006]

Como se muestra, primero tenemos un individuo, que está trabajando en algún dominio, digamos un matemático que trabaja en un departamento de matemáticas, o un pintor en un estudio. El dominio de las matemáticas es todo el conocimiento acumulado sobre las matemáticas. También tenemos bibliotecas de información acumulada. El dominio de la pintura consiste en aquellas obras artísticas del pasado que se exhiben en museos y galerías y se discuten en libros sobre el arte y entre artistas.

Cada subsistema proporciona un importante *momento* en el proceso de creatividad. Robert W. Weisberg explica:

> The domain provides the set of rules and procedures that comprise a specific symbolic context (e.g., the domain of mathematics). Each domain, in turn, is nested in the shared knowledge of a culture. Domains provide patterns of order, or ways of making meaning. For instance, a psychological theory provides a way to think about the self; a musical style provides a way to organize rhythm and melody. These patterns of order expand the reach of our biological existence, but they are not programmed into our genes. [Weisberg, 2006, p. 607]

El enfoque de sistemas de Csikszentmihalyi ve las salidas de un subsistema como entradas a los otros subsistemas. En todas estas transiciones, el tiempo juega un papel esencial.

Según Weisberg, en el modelo de Csikszentmihalyi, la terminología de creatividad sólo debe utilizarse para describir un producto que pasa por todo el ciclo presentado arriba: un producto creativo se convierte en tal sólo después de que se ha convertido en parte del dominio. Es decir, sólo después de que ha sido valorado positivamente por el campo.

Otro aspecto de la creatividad es la reconocida diferencia señalada por muchos entre la creatividad artística versus la creatividad científica. Era una tradición mirar a la creatividad artística como un proceso inherentemente subjetivo y mirar a la creatividad científica como un proceso objetivo que se ocupa de los objetos que existen "afuera", independientes del científico. Sin embargo, tal distinción subjetiva/objetiva no es tan clara como parece.

Weisberg, en su extensa investigación sobre la creatividad y la comprensión de la innovación en la resolución de los problemas, la ciencia, la invención, y las artes, separa dos tipos de pensamiento: e*l pensamiento extraordinario* y *el pensamiento ordinario*.

El pensamiento extraordinario, según él, se construye sobre una conclusión negativa. Sin embargo, un pensamiento ordinario se basa en el pasado y sólo se mueve más allá del pasado a través de pasos incrementales y no es capaz de apoyar los grandes avances que emanan del proceso creativo.

Weisberg examina un subconjunto de la familia de componentes cognitivos que comprenden el pensamiento ordinario: recordar, imaginar, planificar y decidir son los ejemplos de esos componentes.

Weisberg incluye entre las características del pensamiento ordinario las siguientes:

> 1. Our thoughts follow one from another, or are related to one another: Our thinking has structure.
> 2. Ordinary thinking depends on the past: Our thought exhibits continuity with the past.

> 3. Knowledge and concepts direct ordinary thinking: Our thought is directed by top-down processing and exhibits planning.
> 4. Ordinary thinking can be influenced by environmental events: Our thought is sensitive to environmental events.
> [Weisberg, 2006, p. 108]

Un pensamiento ordinario utiliza un subconjunto de los componentes cognitivos del pensamiento con el fin de alcanzar una solución y, por lo tanto, como resultado de esa actividad organizada tiene una estructura y depende de la memoria y la planificación. El pensamiento ordinario es dependiente del pasado por la memoria, también utiliza un procesamiento de arriba hacia abajo, que es el residuo del pasado, y juega roles críticos en la solución de problemas. Sin embargo el proceso creativo funciona para separarse del pasado. Sternberg y Lubart asumen que *la inteligencia analítica*, incluyendo los procesos de la penetración de la codificación selectiva, de la combinación selectiva, y de la comparación selectiva, (según Sternberg & Davidson), desempeña un papel crítico en la reformulación de los problemas. Simonton, basándose en la teoría de Campbell y Poincaré, asume que la combinación aleatoria de ideas, a veces provocada por eventos ambientales fortuitos, es fundamental en la producción de nuevas ideas. [Weisberg, 2006, p. 570]

Cuando se trata de un individuo, es trivial que en el momento que un autor afirma que ha creado una obra de arte, quiere decir que ha creado algo nuevo. Deleuze explica que el único criterio aplicable a cualquier obra de arte es la novedad y continúa: "If you don’t feel you have seen something new, or have something new to say, why write, why paint, why shoot a film?" [Deleuze, p. 220]. Aquí es donde surge la importancia del lector, cuando el campo y la cultura, aparte del autor,

consideran el trabajo como novedoso y creativo. El proceso de llamar "creativo" a algo implica, al mismo tiempo, el proceso de leer / interpretar el trabajo y escribir / interpretar el trabajo en producción. Sin embargo, las interpretaciones de cada lado del trabajo son difierentes entre sí. En el mundo digital, el autor utiliza una semántica diferente, que es la semántica de algoritmos y lenguajes de programación y tiene acceso al código; sin embargo, el lector sólo tiene acceso a *transitoire observable* según Bootz y, por lo tanto, debido a esta brecha semiótica, dos áreas de la textualidad nacen: *text-auteur* y *texte-à-voir*. Explicaré estos conceptos más adelante en este capítulo. Bootz también sugiere una entidad con carácter de actor, a la que llama analista, la cual tiene acceso a ambas semánticas.

C. S. Peirce y su concepción de la novedad proporcionan un marco sistemático para comprender cómo se construyen el pensamiento ordinario y el pensamiento creativo. Peirce, en su fenomenología, define una triada de conceptos que él denomina Primeridad, Segundidad y Terceridad y que corresponden a tres niveles de conciencia. La primeridad pertenece al reino de la posibilidad. Peirce define el sentimiento de primeridad como "an instance of that kind of consciousness which involves no analysis, comparison or any process whatsoever […] it has its own quality which consists of nothing else" [Peirce, p. 1306]. Por ejemplo, una cualidad como lo rojo es una cualidad que consiste en nada más que eso. Lo rojo no existe en comparación con lo amarillo, etc. La segundidad es el modo de ser que está en relación con otra cosa. La segundidad es el nivel de conciencia donde la idea de "realidad" entra en juego. Cuando una manzana encarna la idea de rojo, aparece la

segundidad. Es la categoría que incluye al individuo, la experiencia, los hechos, la existencia y la acción-reacción. Por ejemplo, la manzana que soltamos cae al suelo, o sentimos dolor por un dolor de muelas. Es una relación didáctica, donde se hace un esfuerzo para vincular una cualidad latente con una manifestación de objeto. La segundidad, según Peirce, resulta de una voluntad arbitraria o fuerza ciega (la fuerza de gravedad, en el ejemplo de la manzana), una asociación universal, sin ninguna mediación influyente. En caso de que se trate de una mediación influyente que tiende a mover la dirección de la interpretación, conduce a la terceridad. La terceridad es el mediador a través del cual la primeridad y la segundidad se relacionan. La terceridad es el molde de la mente del intérprete que define el camino tomado entre la primeridad y la segundidad. A pesar de la segundidad que es una categoría de individualidad, la terceridad y la primeridad son categorías de generalidad; pero la generalidad de la primeridad está en el nivel de posibilidad, y la generalidad de la terceridad está en el nivel de necesidad y, por lo tanto, la predicción. La ley de la gravedad, por ejemplo, nos permite predecir que cada vez que tiremos una manzana, caerá al suelo.

Estas categorías construyen los cimientos de las concepciones semióticas de Peirce y, a la vez, están presentes en su concepción de la creatividad. Peirce establece la estructura del significado como una tríada. El proceso de semiosis implica una relación triádica entre un signo o representamen (una idea de primeridad), un objeto (una idea de segundidad) y un interpretante (una idea de terceridad). El representamen es una cosa que representa otra cosa: su objeto. Antes

de que se interprete, el representamen es una potencialidad pura: una primeridad. El objeto es lo que representa el signo. Al interpretarse, el representamen tiene la capacidad de activar un interpretante, que a su vez se convierte en un representamen activando otro interpretante que se refiere al mismo objeto que el primer representamen. Esta triada también desempeña un papel en la clasificación de los signos de Peirce: un representamen puede referirse a su objeto en virtud de la primeridad, la segundidad o la novedad, es decir, a través de relaciones de similitud, contigüidad contextual o ley. Esta tricotomía, es la base para definir un signo en otra tricotomía de icono, índice o símbolo. Un icono es un signo que es un semejante o una representación directa del objeto; como una fotografía que ciertamente se parece a lo que representa. Un índice es un signo que es relativo o insinúa a otro objeto sin semejanza directa; como utilizar una imagen de humo para indicar al fuego. El tercer tipo de signo es un símbolo que es un signo convencional o general que obtiene un significado a través de una conexión habitual y consensuada; como no hay nada inherente en el número 5 para indicar lo que representa.

En la filosofía tradicional occidental, la novedad no puede ser objeto de conocimiento, porque todo objeto de conocimiento debe estar sujeto a las reglas de la razón. La razón entendería la novedad según sus propios esquemas y la reducirá a sus patrones anteriores, por ejemplo, a una lógica de causa-efecto que la desvanecería. Sin embargo, Peirce afirma que la novedad es comprensible, a condición de que se asuma una nueva definición de conocimiento y razonamiento, lejos de cualquier enfoque racionalista (así como irracionalista). Peirce considera tres tipos de

inferencia lógica: deducción, inducción y abducción. La deducción deriva conclusiones lógicas de premisas conocidas universales o supuestas como verdaderas. La inducción viene a construir una conclusión universal a partir de premisas particulares. La abducción comienza con una observación o un grupo de observaciones que presentan una anomalía e intenta encontrar la explicación más sencilla y probable para las observaciones. La deducción es el único razonamiento necesario. Es el razonamiento de las matemáticas. Se parte de una hipótesis, la verdad o la falsedad. La inducción es la prueba experimental de una teoría. Lo único que logra la inducción es determinar el valor de una cantidad. Se establece con una teoría y mide el grado de concordancia de esa teoría con el hecho. Nunca puede dar origen a ninguna idea. Ya no se puede deducir. Todas las ideas de la ciencia llegan a ella por medio de la abducción. La abducción consiste en estudiar hechos y concebir una teoría para explicarlos. Su única justificación es que si alguna vez debemos entender las cosas, debe ser de esa manera [Peirce, p.205]. La presunción, o, más precisamente, la abducción, es el único tipo de razonamiento que proporciona nuevas ideas, el único que es, en este sentido, sintético [Brioschi, 2014, p. 116]. Peirce explica que un hecho debe ser "sorprendente" para activar una abducción, y la fuente de la abducción es un fenómeno sorprendente. Maria Regina Rioschi explica:

> Without the surprising phenomenon, abduction would never begin to occur, but we can determine a surprising phenomenon only through a phenomenological analysis. This implies (i) that the logic of discovery depends upon a previous phenomenological inquiry that offers a surprising phenomenon; (ii) that the precondition of abduction is the presence of a surprising phenomenon. [Brioschi, 2014, pp. 121-122].

Peirce luego sugiere un concepto que él llama "juego puro" como el origen de la sorpresa o el asombro. Él piensa que el juego puro es (1) como "un ejercicio que libera los poderes de uno", (2) sin ningún propósito aparte del de recreación, y (3) siguiendo exclusivamente la ley de la libertad. Peirce afirma:

> It [abduction] begins passively enough with drinking in the impression of some nook in one of the three Universes. But impression soon passes into attentive observation, observation into musing, musing into a lively give-and-take of communion between self and self. If one's observations and reflections are allowed to specialize themselves too much, the Play will be converted into scientific study. [Brioschi, 2014, p. 125]

Por lo tanto, para Peirce, la diversión es a la vez contemplativa, libre e interpretativa.

La concepción del juego puro en Peirce está estrechamente relacionada con la descripción de Wittgenstein de la naturaleza del arte como un juego, y también con la idea de novedad de Whitehead, para quien la novedad es el resultado de un "entretenimiento de posibilidades no expresadas" y se considera la actualización conceptual de posibilidades hasta ahora inexpresadas. [Brioschi, 2014, pp. 209-211]

Los procesos expuestos anteriormente son procesos involucrados en cualquier trabajo creativo. El enfoque de Peirce explica cómo funciona la creatividad para cada entidad del modelo de Csikszentmihalyi. La triada de primeridad / segundidad / terceridad también se puede aplicar a la tricotomía de Csikszentmihalyi sobre la creatividad. Como mencioné antes, un producto creativo se considera creativo cuando se convierte en parte integral del dominio. El autor, desde una primeridad, abduce la creatividad, el campo reconoce la encarnación y crea el vínculo entre el objeto de la

obra y la calidad de "creativo" (segundidad) y la cultura la interpreta como una obra creativa (terceridad). [Brioschi, 2014, pp. 81-96]

II-IV

Pasos en el desarrollo de la ontología

Philippe Bootz, en su ensayo "Towards an Ontology of the Field of Digital Poetry", trata de proporcionar un modelo teórico del campo de la poesía digital basado en la ontología de Spinoza, con notas de Deleuze. Bootz explica:

> Spinoza proposes an ontological model valid for any type of individualised entities called individuals. The individual can thus be a living being, of course, but also an inanimate object like a document or a work. Spinoza opposes eternity and immortality by considering that an individual can change and die. He thus defines the individual using three constitutive dimensions: its physical and material extensive parts, its singular essence and the relationship which links them hic et nunc. An individual can change while adding to his extensive parts of some external elements, but always according to a particular relationship to its essence. The relations between individuals use contacts between extensive parts. These contacts are named “impacts”. An impact provides, for an alive entity, a first emotional experiment, which Spinoza names an “inadequate knowledge. “Adequate knowledge” rests, in addition to the impacts, on relations between the relationships or the essences of the individuals. This knowledge corresponds to scientific knowledge or intuitive knowledge. The death of an individual results in the loss of the relationship between its extensive parts and its essence. [Bootz, 2008, p.1]

En mi opinión lo que entusiasma a Bootz de esta ontología se basa en estos tres factores:

> **Flexibilidad de la ontología:** el individuo puede ser un trabajo o una persona. Por lo tanto, tiene la calidad de la modularidad discutida por Manovich en el capítulo anterior. Un individuo comparte la misma modularidad con el objeto

que en los lenguajes orientados a objetos, o el actor (agente) en el modelo de desarrollo de software basado en agentes. El modelo de desarrollo de software basado en agentes es un paradigma más "metafórico" para conceptualizar sistemas complejos, donde los agentes son abstracciones de entidades vivientes / autónomas del mundo real. Y para Bootz, un individuo existe como un objeto o un agente en estos paradigmas de desarrollo de software. Podría haber varios “instantes” o “estados” de un individuo, cada uno de ellos con diferente configuración del mismo individuo. Esta flexibilidad que origina de la modularidad hace que la implementación de sistemas de software (con un comportamiento autónomo) y la toma de decisiones (en entrono de un modelo conceptual) sean operaciones fáciles y realistas.

Temporalidad: así como un individuo puede morir, una obra digital también muere. Las obras digitales mueren debido a problemas técnicos, como la falta de soporte, la interrupción en el desarrollo de software, la falta de actualización o los cambios extremos de software o hardware.

Distinción de la esencia y la apariencia: la esencia permanecerá. Es decir, los algoritmos o las metodologías utilizadas en un trabajo se mantendrán, pero el trabajo puede morir. Las obras y los individuos se comunican a nivel de apariencia. Sin embargo, sus esencias son diferentes y no pueden comunicarse en ese nivel esencial. Cada esencia puede ser programada en una plataforma diferente.

Bootz también aplica un formalismo gráfico usando un modelo de ensamble y un modelo de entidad / asociación para describir el modelo de Spinoza.

El siguiente diagrama muestra la vida y la muerte de un individuo en el modelo Bootz-Spinoza:

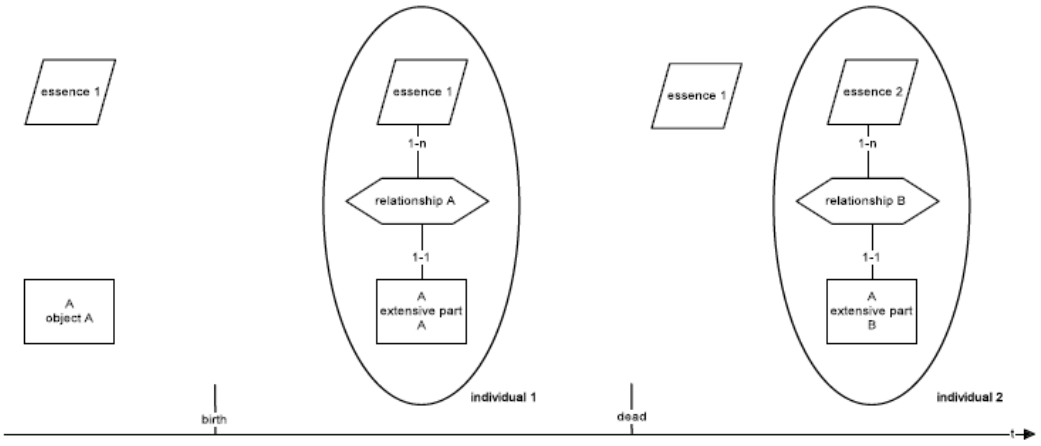

Fig. 18: la vida y la muerte de un individuo en el modelo Bootz-Spinoza [Bootz, 2008]

Antes del nacimiento existen la esencia 1 y la idea del objeto A. El objeto nace cuando la idea del objeto se materializa y aparte de la esencia toma una parte extensa. Cuando el individuo muere, su esencia permanece y su parte extensa desaparece.

Más tarde, Bootz utiliza un modelo de procedimiento para proporcionar un marco que hace posible aplicar el modelo ontológico de Spinoza sobre la poesía digital. En este modelo, la obra, o un poema digital, se compone de extensas partes de material. Algunas partes (programa y datos) están relacionadas con el espacio del autor, y la otra (la observable transitoria) con la del lector. Estos dos espacios están separados y unidos dentro del trabajo, por un dispositivo tecnológico de transformación. Cada actor monopoliza el trabajo de acuerdo con un punto de vista

particular. Los objetos físicos de la obra se integran luego en el dominio del actor de acuerdo con las relaciones interpretativas que el actor puede desarrollar. Estas relaciones son una extensión del actor. Estos actores son roles definidos por los objetivos que identifican sus puntos de vista y las relaciones de contacto que establecen. Cada actor proporciona un "punto de vista" sobre el trabajo. Uno puede imaginar tantos actores como sea necesario, en particular como tipos de analistas (semióticos, históricos, sociológicos, técnicos, críticos, literarios...).

Lo observable transitorio es el resultado perceptible (a menudo multimedia) producido por la máquina mientras se ejecuta el programa. Es, en el lenguaje de Spinoza, una parte extensa del trabajo. El autor crea un programa, pero el lector o el espectador interactúan con un proceso observable que escapa a la voluntad y la lógica algorítmica que el autor ha manifestado en su programa. Ambos, sin embargo, reaccionan a los elementos observables en la pantalla o en cualquier otro medio, cada uno en su esfera, uno en la producción, el otro en la recepción del trabajo. Por lo tanto, el autor crea pero no necesariamente dispone lo que es observado por el espectador/usuario. Los elementos observables por cada uno de ellos difieren porque no son objetos estables y reproducibles, incluso si el autor lo deseara, sino estados transitorios del proceso de ejecución. El resultado observado es generado por un proceso programado, es un estado y no un objeto.

Los siguientes diagramas muestran el esquema estructural y funcional del modelo procesual propuesto.

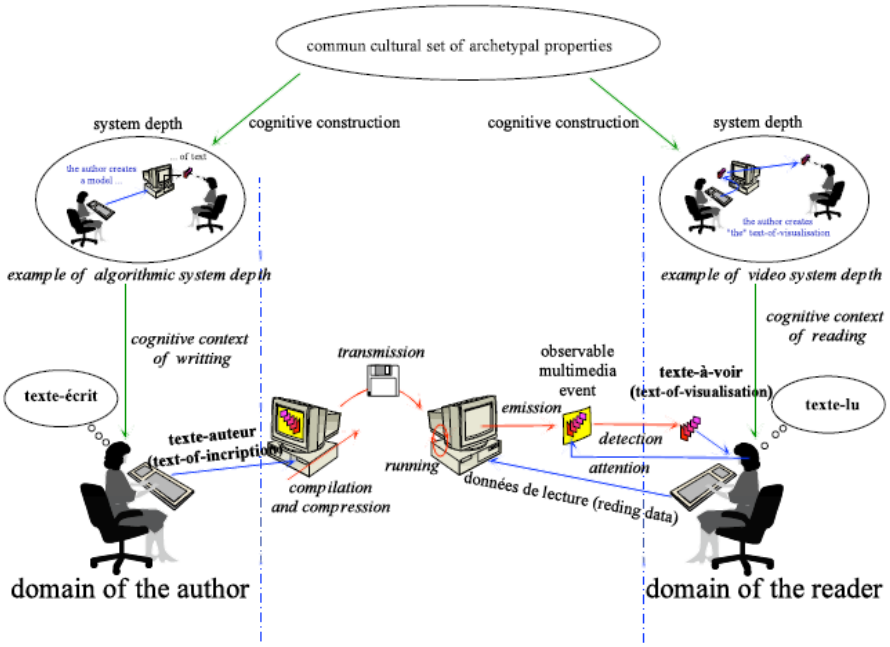

Fig. 19: el esquema estructural del modelo procesual [Bootz, 2008]

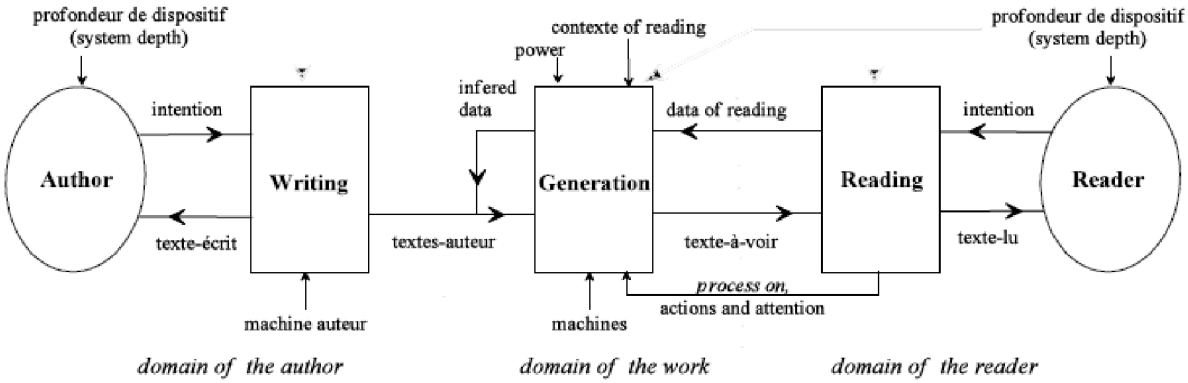

Fig. 20: el esquema funcional del modelo procesual [Bootz, 2008]

Según Bootz:

> In this model, the reader and the author are not people but roles, points of view relate to the work which rest on a possible or prohibited access to the various extensive parts of this one. In particular, the reader does not reach the texte-auteur, the author does not reach the transient observable and none reaches the program while running. Lately the model was growing up by

> adding other roles, in particular roles of meta-readers who are situated across the three quoted domains. A meta-reader reaches the various extensive parts of the work as to the relationships which are established between the reader and the work, therefore with the texte-à-voir. [Bootz, 2008, p.11]

Los pasos mencionados lo ayudan a definir y dibujar una ontología del trabajo digital:

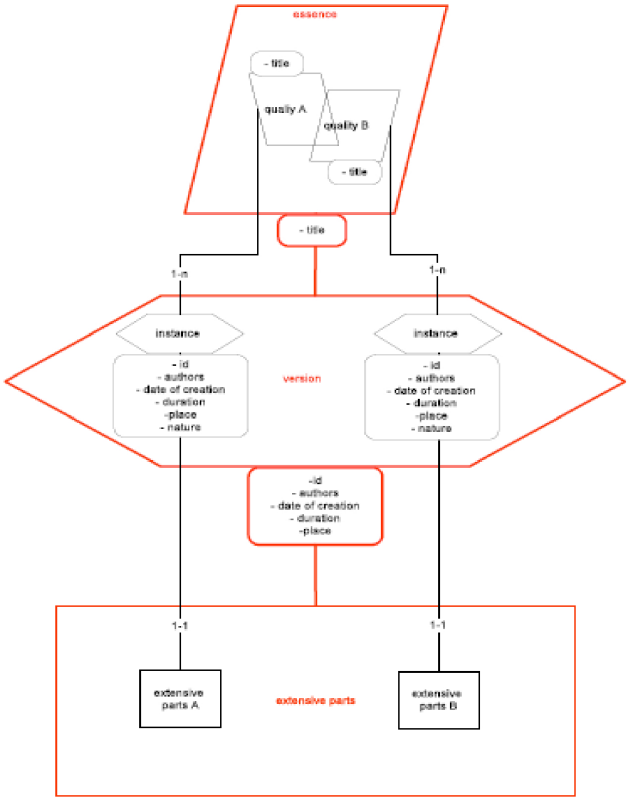

Fig. 21 ontología del trabajo digital [Bootz, 2008]

La esencia está hecha de cualidades, cualidad A y B. Una instancia del trabajo se corresponde con la cualidad A y otra con la cualidad B. Cada instancia tiene una fecha de creación, un autor, etc. Las instancias tienen una parte extensa que se comunica con el usuario y proporciona una interacción con la cualidad correspondiente.

En el diagrama de arriba, la esencia del trabajo es la totalidad de sus cualidades. No es posible acceder directamente a la esencia, y sólo se puede acceder a la esencia mediante el uso de índices como el título del trabajo o algún otro conocimiento que constituye el objeto indicado. La esencia del trabajo se materializa en los objetos de trabajo, en sus partes extensas. El trabajo puede llevar a cabo los procesos de extensión al aparecer en diferentes versiones que pueden ser de diferentes naturalezas: estudio, la versión inicial, actualizaciones, cambio de contexto, etc. Las versiones mantienen entre ellas algunas relaciones de dependencia. Cada una se caracteriza por atributos específicos. Cada objeto, independientemente de esta jerarquía, está en una relación particular con la esencia del trabajo a través de la versión. Una versión determinada se puede rechazar de acuerdo con varias instancias (una instancia en línea, un rendimiento, una instancia en CDR, una publicación en papel, etc.) que son las tantas variantes de la versión. Una instancia es la configuración de trabajo en un contexto dado. Pueden diferir de uno a otro por su método de recepción, su método de difusión o por las tecnologías y programas implementados (reprogramación, simulación). Sin embargo, todas se refieren a las mismas cualidades del trabajo. [Bootz, 2008, pp.13-15]

La ontología sugerida por Bootz para la poesía digital se centra principalmente en la actividad de clasificación de los individuos. Su sugerencia puede ser expandida por varias otras ontologías. Bootz afirma:

> This largely open project of indexing needs a broad multidisciplinary cooperation to be concluded because the definition of ontologies of actors must be a collective project. It could be carried out within the framework of a European project. [Bootz, 2008, p.25]

La ontología de Bootz consiste en una ontología principal de la obra, una ontología del actor y una ontología del documento. Pero cada una de estas ontologías mencionadas también consiste en una red de otras ontologías. Por ejemplo, para una ontología completa de todos los actores, debemos tener ontologías separadas para un historiador o un crítico literario. Estas diferentes pequeñas ontologías se unen y crean una red de varias ontologías que juntas pueden explicar el proceso de lectura o escritura de un poema digital.

Desafortunadamente, este proyecto colectivo de definición de ontologías para cada actor, cada documento y cada tipo de obra nunca se realizó debido a la falta de recursos y al extenso proceso de desarrollo de la ontología. Sin embargo, discutiré la versión preliminar de Bootz y trataré de ampliarla y transformarla en un nuevo modelo que sugeriré.

Un poema digital que contiene software implica la presencia del proceso de desarrollo de software como un proceso inseparable de la escritura del poema digital. Sin embargo, la informática y la escritura creativa no comparten un vocabulario común. Por lo tanto, para crear una ontología más consistente, una traducción de taxonomías parece obligatoria.

Considerando los tres agentes de la creatividad representados arriba, una traducción de las taxonomías podría ser obtenida equiparando a la “cultura” con el conocimiento del dominio de la ingeniería de software; al “campo” con el

conocimiento del subdominio de la poesía digital y al “individuo” con el conocimiento de la instancia de la IS (Ingeniería de Software).

II-V

Ontología del poema digital

En la ontología Bootz-Spinoza, la esencia del trabajo es accesible a través del desarrollo de conceptos. Un agente, por ejemplo un analista, se pondrá en contacto con la parte extensa del trabajo y proporcionará un documento sobre el trabajo. El documento consta de secciones. Un documento físico se divide en secciones, cada una de las cuales se puede atribuir a un solo rol (un actor), es decir, desarrolla un solo punto de vista. Por ejemplo, en un documento académico, las secciones que expresan una actividad de lectura son fácilmente distinguibles de aquellas que expresan una actividad de análisis, por ejemplo, en esta tesis hago una lectura de la historia de los lenguajes de programación y realizo un análisis de la poesía generativa. Cada una de estas partes corresponde a actividades diferentes: una de lectura y otra de análisis.

Por lo tanto, cualquier trabajo tiene instancias y cada instancia del trabajo es accesible en sus partes extensas para un actor que observa el proceso de lectura o escritura. Y también, cualquier instancia del trabajo es indexable a través de un documento que consta de secciones en el que cada actor desarrolla conceptos desde el punto de vista de cada sección.

De Bootz tomo prestado su concepto de actor, documento y trabajo -que usaré-, sin embargo, su enfoque hacia el campo de la poesía digital es útil para la clasificación y la documentación del campo y no para comprender el proceso creativo del campo. Su proyecto puede producir ontologías extensas y puede documentar los

trabajos producidos en el campo, y después de una documentación tan extensa, un analista puede llegar al análisis cuantitativo y cualitativo del campo.

Las discusiones y los conceptos introducidos en la tesis hasta ahora nos permiten sugerir las siguientes afirmaciones que definen la ontología de un poema digital:

1. Un individuo consiste en una esencia y una parte extensa.

2. Los individuos interactúan con sus partes extensas y ninguno de ellos tiene acceso a la esencia de otro.

3. Un individuo puede ser una obra o un actor.

4. Cuando una obra es llamada “objeto creativo” por al menos un actor, la obra es un poema.

5. Un documento es el resultado del contacto entre el actor y la obra.

6. Un documento consta de secciones.

7. Una sección es una parte extensa de un actor debido al contacto que el actor tiene con la obra.

8. El contacto que un actor establece con la obra es de tres tipos: escribir, leer y analizar. Por lo tanto, el actor que produce la sección correspondiente es el escritor, el lector o el analista.

9. Un lector o un analista puede ser uno de los tres: una persona, un dominio o una cultura.

10. La sección de la obra producida por el escritor se llama *text-auteur* y la sección de la obra producida por el lector se llama *texte-à-voir*.

11. *Text-auteur* es inaccesible para el lector y *texte-à-voir* es inaccesible para el escritor, pero ambos son accesibles para el analista.

12. Un poema digital es un poema que contiene un software.

13. Un software es un conjunto de datos o instrucciones de computadora o programas que le dicen a la computadora cómo funcionar.

14. Una computadora es un artefacto cognitivo que consiste en software y hardware, capaz de memorizar, interpretar, buscar y conceptualizar. El límite de cada capacidad mencionada depende de las especificaciones de software y hardware.

15. Un software es producido por los actores que se ponen en contacto con la obra en las siguientes secciones: la investigación, la planificación, el diseño, el desarrollo, la prueba, la instalación y el mantenimiento. Cada una de estas secciones se produce con un lenguaje diferente.

16. Un lenguaje es un sistema de signos y de relaciones entre ellos que es comprensible por un artefacto cognitivo o por una persona.

17. La obra puede realizar procesos de extensión apareciendo en diferentes versiones que pueden ser de diferentes naturalezas: estudio, versión inicial, actualizaciones, cambio de contexto, etcétera.

18. Cada versión implementa propiedades específicas de la obra, aunque quizás no todas al mismo tiempo, en los objetos (archivos, *observables*

transitorios, programa fuente, situación de lectura y dispositivos en el caso de actuaciones.)

19. Una instancia es la configuración de la obra en un contexto dado. Puede diferir de uno a otro por su método de recepción, su método de difusión o por las tecnologías y programas implementados (reprogramación, simulación). Pero se refiere a las mismas cualidades de la obra, es decir, están en la misma relación (la versión) con la esencia de la obra, mientras que las versiones pueden desarrollar diferentes cualidades. Por ejemplo, Google Chrome, como navegador, puede tener instancias desarrolladas para teléfonos móviles o computadoras personales. Una versión antigua del mismo programa tiene menos utilidades que una nueva versión: menos calidad gráfica o menos posibilidades de interacción.

20. Una obra muere cuando su parte extensa pierde el contacto con su esencia. La parte extensa puede unirse a la parte extensa de otra obra viva.

21. Una obra se suspende cuando ningún actor aparte de su autor establece contacto con su parte extensa.

En las partes siguientes de este capítulo utilizaré un enfoque interpretativo sobre tres tipos de poesía digital con el fin de extender mi ontología (que es diferente de la de Bootz), y la usaré en el último capítulo para realizar una lectura minuciosa en que analizo poemas digitales. La poesía generativa se basa en una codificación y una concepción predefinida de la poesía. Es el tipo más antiguo de poesía digital. El

hipertexto usa una codificación mínima y estructuras predefinidas, y es uno de los tipos más populares de poesía digital. La poesía de Twitter es el resultado de un conjunto de redes sociales y poesía generativa.

Ontologías específicas

1. Poema generativo

Charles Sanders Peirce introduce una distinción entre el tipo y el token. La *Enciclopedia de filosofía de Stanford* define esta distinción como sigue:

> The distinction between a type and its tokens is an ontological one between a general sort of thing and its particular concrete instances (to put it in an intuitive and preliminary way). […] Types are generally said to be abstract and unique; tokens are concrete particulars, composed of ink, pixels of light (or the suitably circumscribed lack thereof) on a computer screen, electronic strings of dots and dashes, smoke signals, hand signals, sound waves, etc.

Lo que aparece en el papel o en la pantalla siempre es un token, no un tipo. Sin embargo, la materialidad de los tokens es diferente. Una palabra en sí misma es un tipo en el que pueden existir varios tokens. En este sentido, las palabras almacenadas en la base de datos de un poema generativo son tipos y cada aparición de ellas en cualquier poema generado es un token.

Dentro de la misma estructura comparativa, la esencia de un poema digital (una obra), consiste en un contenido y algunas formas y estructuras abstractas a las

que llamamos diseño y que pueden tener varias instancias a través de diferentes implementaciones. *Patchwork Girl*, por ejemplo, tiene una estructura hipertextual y un contenido, y tiene instancias que son accesibles a través de sus partes extendidas por un entorno StorySpace en una PC, o por el navegador web Mozilla en una tableta.

Utilizo la distinción token / tipo cuando me refiero a las palabras o los signos en general y utilizo la distinción instancia / esencia al referirme a la implementación / diseño.

Aunque expresé en la sección anterior que un trabajo es un poema cuando un lector lo llama *poema*, al escribir, un autor debe tener una definición de poesía, la cual es una mezcla de definiciones positivas y negativas de lo que es poesía y de lo que no lo es. En la poesía generativa, una definición ampliamente aceptada en el campo pertenece a Manurung. Este autor acepta que no existe una definición clara y válida de lo que se puede considerar como poesía, pero ofrece una definición positiva que consta de tres atributos:

1. Significatividad: El texto de un poema debe transmitir intencionalmente algunos mensajes conceptuales que son significativos bajo algunas interpretaciones.
2. Gramaticalidad: el texto de un poema debe obedecer las convenciones prescritas por una gramática y un léxico dados. Sin embargo, debido a la

importancia de la licencia poética, este atributo se rige por las reglas del lenguaje figurado.

3. Poeticidad: un poema debe exhibir características poéticas tales como patrones rítmicos y rima. [Manurung, 2004, p. 15]

Por lo tanto, de acuerdo con Manurung, un poema es un artefacto del lenguaje natural que cumple simultáneamente las propiedades de significatividad, gramaticalidad y poeticidad. En su análisis, sugiere que los generadores de poesía basados en plantillas pueden producir textos que son capaces de cumplir sólo con la gramaticalidad. Los generadores de texto con reconocimiento de formularios pueden alcanzar la poeticidad y la gramaticalidad. Y también hay otra categoría de generador de poemas que puede cumplir todas esas tres propiedades.

Sus diferencias radican en su definición de poesía y también en el algoritmo que ha sido diseñado para la generación de texto.

Anteriormente, mencioné las categorías que Carolyn Lamb sugirió para el software de generación de poesía. Manurung propone una categorización ligeramente diferente:

Generadores de sopa de letras. Estos sistemas no intentan cumplir ninguno de los atributos mencionados y simplemente concatenan palabras aleatorias. Por ejemplo, el poema llamado "LYRIC 3205" de Pete Kilgannon:

judy gotta want upon someone.
wanna sadly will go about.

sammy gotta want the thief him but the
every reason. real distance carry.

Generadores basados en plantillas y gramática. Se trata de un enfoque un poco más sofisticado que el anterior. La generación de estos sistemas se lleva a cabo seleccionando aleatoriamente las palabras de un léxico y utilizándolas para llenar los espacios en blanco de las plantillas de oraciones predefinidas. Las palabras y los blancos se clasifican como partes del discurso y por lo general sólo se trata de palabras y no de funciones; es decir, son sustantivos, verbos, adjetivos y ocasionalmente adverbios, que pueden variar. Por lo tanto, se puede decir que estos sistemas satisfacen, aunque de manera muy limitada, la propiedad de la gramaticalidad.

Generadores de texto con reconocimiento de formularios. Estos generadores intentan explícitamente cumplir las propiedades de gramaticalidad y poeticidad, y específicamente la de métrica. Utilizan algoritmos más sofisticados y sistemas más desarrollados para llenar sus plantillas. Por ejemplo, *Rimbaudelaires* es un programa generado por el grupo ALAMO. El programa usa la poesía de Rimbaud y de Baudelaire. Cambia las palabras en sonetos de Rimbaud con palabras seleccionadas de la poesía de Baudelaire.

Sistemas de generación de poesía. Estos generadores intentan explícitamente cumplir con las tres características mencionadas. Utilizan algunas

técnicas generales de resolución de problemas aplicando principios de inteligencia artificial (IA), como el razonamiento basado en casos. Un razonamiento basado en casos intenta resolver un nuevo problema consultando una base de datos explícita de problemas existentes y sus soluciones. Estos generadores al mismo tiempo pertenecen a un subcampo de la IA llamado "Generación de lenguaje natural" o NLG. [Manurung, 2004, pp.18-29]

Cualquier sistema de NLG consta de cuatro elementos: 1) Una fuente de conocimiento -intertextualidades, por ejemplo, poesía de Cervantes-; 2) Un objetivo comunicativo -un lenguaje sofisticado, o una jerga de lenguaje-; 3) Un modelo de usuario de una audiencia prevista -lectores expertos de literatura o usuarios generales de internet-;y 4) un modelo de discurso -poesía, ficción, etc.-

Los sistemas clásicos de NLG utilizan la arquitectura de tuberías de la siguiente manera:

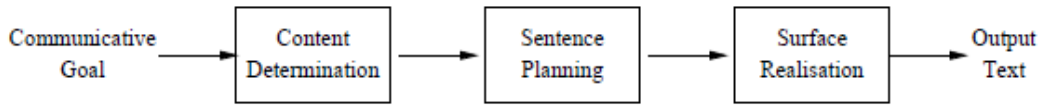

Fig. 22: arquitectura de tuberías en sistemas clásicos de NLG [Manurung, 2004]

En este diagrama, la determinación de contenido trata con la semántica proposicional del mensaje. Una vez que ha seleccionado un conjunto de proposiciones, se compromete con esta selección y pasa a la fase de planificación de la oración, donde las estructuras sintácticas se seleccionan y arreglan para transmitir las proposiciones elegidas. Dentro de la fase de realización de la superficie, se consultan los recursos

lingüísticos y las estructuras sintácticas se transforman en cadenas de texto. [Manurung, 2004, pp.32-46]

Existen diferentes enfoques para mejorar el modelo clásico mediante el uso de una arquitectura alternativa o el empleo de mejores metodologías de búsqueda. En esta etapa, la calidad literaria de un texto producido depende completamente del algoritmo utilizado en el programa y del paradigma de diseño del software.

La descripción que proporcioné aquí sobre poesía generativa puede leerse en paralelo con el concepto de Weisberg sobre pensamiento ordinario/extraordinario: la poesía generativa tiene una estructura y sigue algunas formas preexistentes. También depende del pasado; todos los textos provienen de un dominio de conocimiento que existía antes del poema generado. El conocimiento y los conceptos dirigen la generación de los poemas. Es decir, la poesía generativa cae principalmente en la categoría del pensamiento ordinario y no en el pensamiento creativo o extraordinario. Sin embargo, hay elementos que separan la creatividad de máquina de la creatividad humana. Es decir, un poema generativo puede ser creativo debido a los siguientes elementos:

> Una máquina no puede estresarse ni sentirse amenazada ni cansarse. Weisberg menciona que el pensamiento ordinario puede verse influenciado por acontecimientos ambientales. Los únicos sucesos ambientales que pueden influir en la generación de un poema por parte de su programa, están relacionados con el hardware de la computadora o con el sistema operativo.

Sin embargo, cuando un programa está funcionando, no es sensible a ningún suceso ambiental. Yo llamo a este factor *la característica libre de ambiente*.

La capacidad de memoria, en un poema generativo, es decir, la capacidad de intertextualidad dentro del poema generado, depende de las bases de datos o de los corpus que funcionan como sus fuentes de conocimiento. Teóricamente, éstas son ilimitadas. Este no es el caso de la memoria humana. Es decir, teóricamente un poema generativo puede extender su memoria del pasado a toda la historia digitalizada de la literatura. Yo llamo a este factor *la extensión del pasado.*

No sigue las convenciones estéticas detalladas. Una metáfora no puede ser agradable o desagradable para una máquina. Por lo tanto, puede crear metáforas o imágenes que pueden parecer contra-estéticas. En este aspecto, un poema generativo sin ninguna intencionalidad propia puede ir hacia una conclusión negativa, una noción que Weisberg califica como un atributo del pensamiento extraordinario. Es decir, a pesar de que un poema generativo se produce de acuerdo con tres de los cuatro componentes del pensamiento común -me refiero a recordar, planificar y decidir- no puede imaginar. La falta de imaginación, al mismo tiempo, significa que es posible imaginar todo. Llamo a este factor *paradoja de la imaginación*.

Por estas razones, un poema generativo, aunque sigue los principales componentes de un pensamiento ordinario, también puede producir poemas que aspiran a un

pensamiento extraordinario. Se puede utilizar los factores creativos de un poema generativo como un borrador para la creatividad humana, como el ejemplo de Leevi Lehto que mencioné antes en el que este autor utilizó los poemas generados de su programa Google Poetry Generator como la versión borradora del libro que publicó después, revisando los poemas generados.

La ontología del poema digital se puede extender al agregarle las siguientes líneas, para definir la ontología de un poema generativo:

> 22. Un poema generativo es un poema digital que contiene un algoritmo generativo en su software capaz de producir los siguientes efectos en lenguajes naturales: significatividad, gramaticalidad y poeticidad.
>
> 23. Un algoritmo generativo tiene cinco categorías: generadores de sopa de letras, generadores de texto, generadores de texto con reconocimiento de formulario, generadores de búsqueda y composición o generadores de lenguaje natural.

He mencionado antes la estructura de jGnoetry del corpus que estoy analizando. Ahora, doy una explicación más detallada sobre otros dos poemas generativos del corpus: PoetRNN y Poem: RNN. La palabra RNN en estas obras se refiere a las redes neuronales recurrentes (Recurrent Neuronal Network). Este tipo de redes neuronales, trata de simular la memoria humana y la forma en que aprendemos. La red realiza la conexión entre los objetos que observa y almacena la observación cada vez en una

celda de memoria. De nuevo, realiza observaciones adicionales y corrige sus resultados pasados. Este proceso de observación y corrección de la hipótesis se denomina entrenamiento. La lógica detrás de este tipo de red neuronal es abductiva en términos Peircianos. Varios científicos informáticos reconocieron el valor de este tipo de red para imitar el comportamiento humano. Se utilizó con éxito en el reconocimiento de escritura manual o reconocimiento de voz. Los experimentos realizados con ella también demostraron que puede aprender la gramática del lenguaje natural con aproximadamente el 93% de precisión o más, dependiendo de la cantidad de objetos (aquí corpus textual) observados y analizados. Tanto poetRNN como Poem: RNN aplicaron este tipo de red neuronal para ser entrenados con las obras de los grandes poetas como Shakespeare, Pushkin, etc. Sin embargo, tienen algunas diferencias. Poem: RNN aprende observando caracteres en un texto y PoetRNN observando palabras en un texto. Debido a que el número de letras en el alfabeto humano es limitado, Poem: RNN aprende la gramática más rápido y produce poemas que son más comprensibles. En cambio, PoetRNN necesita un corpus textual original muy grande para poder producir poemas con sentido. En ambos casos, el lenguaje usado en los poemas depende del corpus original. Es una imitación o una simulación de la poesía del poeta con el que fueron entrenados.

Por ejemplo, entrené a Poem: RNN con la poesía de García Lorca y el siguiente es un poema que generó el programa:

Estoy en los campos de las manos del alma, sin embargo, sobre lo almacional del mundo. En el camino, el cielo es el agua que es el aire, la ciudad es un poco de perdonares, ante el mundo, le podían, sin salir de su compañera, el campo se extienda a su mejor de una calle, y el agua se encontraba en el agua de la mano, los corredores del agua en la ciencia, en el aire de la madrugada, lleno de piedra, entre la calavera, es el amor,

la puerta de un mar de agua estaba suspirado por la ciudad, la misma construcción, la mano alta en la palma de lo alto.

La punta del mar se desliza entre la madrugada. Algado de las piernas y sombras estaba alli desde el mar.

Al mar.

En el caso de PoetRNN, pude producir haikús como este:

rain pond

the light of shadow

of the song

not the star

the scare of the crocks

of the frost

El programa PoetRNN puede producir poemas con más sentido si se entrena en una computadora mejor que mi computadora personal, con un corpus más extenso. Los algoritmos que utilizan estos dos programas son muy similares, pero las estrategias que toman para el entrenamiento son diferentes. Además, los poemas generados por

el algoritmo RNN tienen una estética diferente a los poemas generados por las restricciones de OULIPO como Oulipoems. El algoritmo claramente juega un papel determinante en su significado.

2. Poema de Twitter

La poesía de Twitter es un tipo de poesía generativa que utiliza el entorno de redes sociales de Twitter como su espacio creativo. Se trata de un espacio en el que se presenta el poema y, al mismo tiempo, se reescribe dicho poema a través de las interacciones y participaciones del lector. En Twitter, cada publicación es pública y esa es la razón por la que un autor puede llegar a un lector completamente desconocido para él o para sus conocidos. Además, hay una restricción de 140 palabras para cada publicación. Esta restricción, para los poetas de Twitter, funciona como una limitación oulipiana. Por ejemplo, *Everyword* es un poema de Twitter publicado en el tercer volumen de la antología ELO. El poema fue escrito en Python por Allison Parrish. Este poema obtuvo 100,000 seguidores, entre 2007 y 2014. El poema sólo publica una palabra del diccionario cada 30 minutos. Una simple palabra del diccionario no puede formar un poema por sí solo. Sin embargo, la idea creativa del poema reside en la interacción que la palabra recibe de los lectores.

Fig. 23: Everyword de Allison Parrish en Twitter

Los lectores pueden comentar, compartir la palabra, expresar si les gusta la palabra y también pueden invitar a otros lectores a participar en la reflexión sobre la palabra. Todas las interacciones de los lectores se guardan y forman un nuevo texto que se puede interpretar desde varios puntos de vista: literario, social, político, etc. Además, si a algunos usuarios de las redes sociales el poema les parece molesto o acosador, pueden denunciarlo como un abuso, y si el número de lectores que denuncian el abuso en un poema es alto, la obra puede eliminarse y, en consecuencia, se puede eliminar al autor y su existencia del medio.

Por lo tanto, al definir la ontología de un poema de Twitter es importante tener en cuenta todos estos elementos. Aquí está el rol que podemos agregar a nuestra ontología principal para que cubra los poemas de Twitter:

24. Un poema de Twitter consiste en un poema generativo creado con la restricción oulipiana del uso de 140 palabras, además de otras interferencias textuales de los actores de un entorno digital llamado Twitter, que representa una entidad de actor que llamamos poder.

25. La interferencia textual de los actores puede ser de expresión, reflexión, protesta, deseo de compartir o comentar el poema generativo.

26. El poder puede desconectar la relación entre la esencia y la parte extensa de un poema de Twitter. Esto puede ser desencadenado por la protesta de otros actores o por la decisión arbitraria del mismo poder.

3. Poema de hipertexto

Introduje la poesía hipertextual en el capítulo anterior a través de las definiciones de Landow y Aarseth. Resumo las definiciones introducidas ahí para nuestra ontología de la siguiente manera:

27. Un poema de hipertexto es un conjunto de poemas digitales en el que cada poema está en relación con otro poema, a través de una materialización de esta relación en una conexión llamada enlace.

28. Los poemas digitales que forman un poema de hipertexto y los enlaces entre ellos pueden generarse, reemplazarse o seguirse en respuesta a las interacciones de cada actor. Los enlaces se pueden programar para que sean

axiales (para tener un orden lineal), arborescentes (para hacer un orden de estructura de árbol), conectados en red (para mantener un orden de estructura de gráficos) o en capas (para combinar dos capas de enlaces en la red: una con una dimensión de tiempo y otra sin él).

29. El objeto del poema, en un poema de hipertexto, puede componerse de texto, imagen, sonido o enlace.

III

La codificación creativa: un estudio de caso

A well-written program is its own Heaven;
a poorly-written program is its own Hell.
– Tao of Programming, Geoffrey James

Como comentamos en capítulos anteriores, cada texto de poesía digital se compone de al menos tres textos diferentes o, en otras palabras, está destinado a tres subjetividades diferentes: código de máquina que está destinado a ser interpretado por la máquina, lenguajes de programación que están destinados a ser entendidos por el programador y la máquina, y el programa en acción que está destinado a ser entendido y utilizado por la mente humana. Si para Peirce mente y subjetividad son materia de la semiótica, entonces, el poema digital está sujeto a tres semióticas distintas. Cada una de estas semióticas se compone de diferentes sistemas de signos.

La subjetividad de una máquina está compuesta por un sistema de signos que está en relación con sus elementos de memoria, procesadores, mecanismos de entrada y salida, y su arquitectura específica. La subjetividad de un lenguaje de programación, sin embargo, se compone de una relación dialéctica entre la subjetividad humana y la máquina y, aunque la máquina ejecuta (o instancía) el programa, no lo entiende por medio de signos comprensibles para la subjetividad humana. Por otro lado, el lector humano comprende los signos dentro de una

semiótica ajena al 0 y el 1 de la máquina. Además, tanto el lenguaje de programación como el de la máquina están sujetos a cambios radicales, ya que su anatomía puede cambiar con el tiempo. Un programa escrito en un paradigma declarativo o por un paradigma imperativo toma diferentes enfoques interpretativos para el programador y la máquina.

Tanto Eco como Colapietro parten de la teoría semiótica de Peirce. En su teoría general del signo, Eco destierra la subjetividad de la semiótica. Sin embargo, Colapietro adopta un enfoque diferente del de Eco que puede darnos una posibilidad de expandir la semiótica uni-signo (donde un solo sistema de signos produce el significado) a una semiótica poli-signo (donde varios sistemas de signos generan un signo híbrido que produce el significado). Si vamos a avanzar hacia una semiótica de la creatividad en la poesía digital, no podemos descuidar el impacto de estas tres categorías diferentes de usuarios del signo y, por lo tanto, no se puede tener en cuenta un enfoque como el de Eco.

En *Peirce's Approach To The Self* , Colapietro afirma:

> Eco stresses the importance of the notion of interpretant for an understanding of signs; moreover, he recognizes how insightful is Peirce's suggestion that both actions and habits of action can function, though in different ways, as interpretant of signs. Yet he fails to see the implications of all this for a semiotic treatment of the human subject. [Colapierto, 1988, p. 45]

El enfoque de Eco tendría varios efectos limitantes en un análisis semiótico del objeto creativo digital. Por ejemplo, los hábitos de acción juegan un papel importante

en el proceso de programación. El hábito de un programador es parte inseparable de su forma de pensar donde el cuerpo del programador se integra con su manera de pensar, así como la configuración de una máquina determina los hábitos con los que actúa. Empero en esta tesis no vamos a analizar los hábitos de programación en las piezas de arte digitales; ese sería, por sí solo, tema de otra tesis. Para tener una idea más clara sobre los hábitos de los programadores, puedo mencionar que cada programador define una configuración personal para sus atajos de teclado. Algunos programadores son diestros y otros zurdos, por lo que cambian el orden de las teclas para copiar/pegar, etc. Estas configuraciones cambian entre programadores incluso de un lenguaje de programación a otro, o de un IDE a otro, etc.

Al referirse a Peirce, Colapietro explica: "La persona y el signo se educan recíprocamente" [Colapierto, 1988, p. 308] Los letreros informáticos han cambiado radicalmente desde la era de las tarjetas perforadas hasta nuestras computadoras modernas, eso se debe a que los tres agentes, el programador, la máquina y el lector/usuario han interactuado recíprocamente con el signo y se han educado mutuamente.

Eco afirma que "la cultura se puede estudiar completamente bajo un perfil semiótico". Al comienzo de *Una teoría de la semiótica* expresa que un semiótico tiene derecho a tratar la cultura como un sistema de signos, siendo los signos aquellos fenómenos que dependen de una compleja red de códigos superpuestos y producidos por el esfuerzo del trabajo [Colapierto, 1988, p. 309]. Colapietro desafía esta afirmación:

> Semiotic explorations of human subjectivity do not illicitly pass beyond the boundaries of semiotics; rather such explorations naturally push outward these boundaries. [Colapierto, 1988, p. 45]

Lo mismo podría decirse de la “subjetividad” de la máquina. Tratar a la máquina como un texto, sin estudiar sus variadas peculiaridades reduciría la semiótica, fundamentalmente híbrida, a los signos comprensibles para humanos, que es el camino tomado por Aarseth y otros teóricos en el campo. Una generalización que elimine a la máquina de la semiótica se quedará corta cuando intente explicar los aspectos cognitivos y culturales del lenguaje de la máquina. Colapietro explica:

> [For Peirce] the subject is seen not as a primordially free source of thought and action, but rather as a being so deeply embedded in its time and place as to be largely, though not completely, constrained in its cognition and conduct [...] For even clearer evidence of Peirce’s awareness of the cultural overdetermination of the human subject, consider the following passage: “A great many people think they shape their lives according to reason, when it is really just the other way [around]. [Colapierto, 1988, p. 40]

Ravi Vatrapu y Dan Suthers reconocen dos estilos en las comunicaciones culturales: alto contexto y bajo contexto. Las comunicaciones de alto contexto apuntan a las emociones y las persuasiones retóricas, su discurso es largo y sin prisas, no tienden a

transmitir información porque la información se incorpora junto con el contexto; el envío y la recepción de su mensaje tienen como efecto el disfrute; su ambigüedad es permitida. Sin embargo, las comunicaciones de bajo contexto se centran en el uso de información racional, por lo que en ellas la información tiene una cantidad definida y debe entregarse de una vez [Vatrapu, 2007, p. 8]. Aquí, la información sustenta decisiones, los errores conllevan costos y difuminan la propia información, y debe evitarse la ambigüedad. En un poema digital sólo la presentación final del poema utiliza un estilo de comunicación de alto contexto y las otras dos presentaciones del poema (el texto de la máquina y el código fuente) se comunican en un estilo de bajo contexto, es decir, pertenecen a diferentes categorías culturales. El discurso de los estilos de la comunicación nos da un contexto donde el individuo o el actor trabajan de ida y vuelta con los signos entre su sistema cognitivo y el de la cultura.

En una dimensión cognitiva de la cultura, el enfoque tradicional de la psicología dominante hace cuatro supuestos básicos sobre la cognición:

1. Universalidad, en la que la sensación, la atención, la percepción y la memoria son universales e invariables en todas las culturas.
2. Independencia de contenido, en la que los procesos cognitivos básicos son invariables a lo largo del contenido.
3. Suficiencia ambiental, en la que el entorno aporta contenido al proceso cognitivo sin necesidad de intervenciones culturales o sociales.

4. Varianza cultural infinita, en la que el rango de las culturas es una función de la varianza en las condiciones ambientales. [Vatrapu, 2007, p. 8].

Estos supuestos conducen a una disociación fundamental entre la cognición y la cultura. La varianza cultural infinita no sería posible si la cultura y la cognición no fundamentalmente disociadas por la universalidad de la cognición (independientemente de la cultura) o la independencia de la cognición del contenido o su suficiencia ambiental. Sin embargo, en un enfoque más reciente inspirado por la evolución de la WWW, Nisbett y sus colegas de la psicología transcultural dicen que la cultura y la cognición interactúan. Por ejemplo: algunas culturas prestan más atención a los objetos individuales y otras a los campos perceptivos. Algunas culturas tienen una percepción orientada a objetos y otras están orientadas a las relaciones. Algunas organizan el conocimiento de manera categorial y otras de manera relacional y algunas razonan analíticamente y otras de manera integral. Este enfoque se puede rastrear en la forma en que un programador latinoamericano o del este de Asia escribe código de manera diferente a la de un programador europeo.

A continuación trazaré una semiótica híbrida de la poesía digital compuesta por tres semióticas: la semiótica de la máquina, la programación y los lenguajes naturales, en los que el lenguaje de programación actúa como puente entre la máquina y la subjetividad humana.

Explorar las dimensiones semióticas o realizar un estudio sobre la subjetividad del lenguaje de máquina es una investigación que está en manos de los

científicos de la computación y el estudio de la semiótica de los lenguajes naturales ha sido emprendido por varios otros investigadores y pensadores en humanidades. Este capítulo sólo se enfocará en el puente entre estas dos subjetividades con respecto a unas piezas seleccionadas de poesía digital.

III-I

La subjetividad de la máquina

La mente humana escribe el programa, luego, eso se traduce a código de máquina, sin embargo, un código de máquina no es fácilmente comprensible para la mente humana[6]. En realidad, el código de máquina es tan ilegible para los humanos que la Copyright Office of the United States no puede identificar si un código escrito en lenguaje de máquina es original o una copia. En la última década, los sintetizadores semánticos son utilizados para responder preguntas sobre las propiedades de los dígitos binarios. Srinivasan explica:

> Semantics-based binary-rewriting tools become particularly important when one wishes to modify the functionality of the binary, and the source code and/or compiler toolchain for the binary is unavailable. Semantics-based binary rewriting can be done for the purposes of binary optimization (offline optimization, superoptimization), software reuse (partial evaluation, slicing, binary translation), or software security (binary obfuscation, de-obfuscation) [...] A key challenge in synthesizing machine code is the enormous size of the synthesis search-space. Instruction sets like Intel's IA-32 have around 43,000 unique instruction schemas; this huge instruction pool, along with the exponential cost inherent in enumerative synthesis, results in an enormous search space for the synthesizer: even for relatively small specifications, a naïve synthesizer might take several days to find an implementation. [Srinivasan, 2017, p. 12]

En palabras menos técnicas, una máquina actúa en términos de memoria, procesador, entrada y salida. El tamaño de la memoria de la máquina define la medida en la que esta puede recordar instrucciones previas, declaraciones, información, etc. Un procesador define la velocidad a la que una máquina puede analizar, interpretar o

[6]https://www.cise.ufl.edu/~mssz/CompOrg/CDA-lang.html

realizar una acción. La entrada y la salida definen el dominio cultural en el que reacciona una máquina. Un teléfono celular es una máquina que reacciona en el ámbito cultural de la comunicación cotidiana. Un reloj inteligente es una máquina que actúa en el ámbito cultural de la programación diaria y la salud. Una computadora personal, un reloj inteligente o un teléfono celular son computadoras diferentes que usan una semiótica similar pero actúan en diferentes discursos o dominios culturales y no pueden separarse fácilmente de ese discurso o dominio.

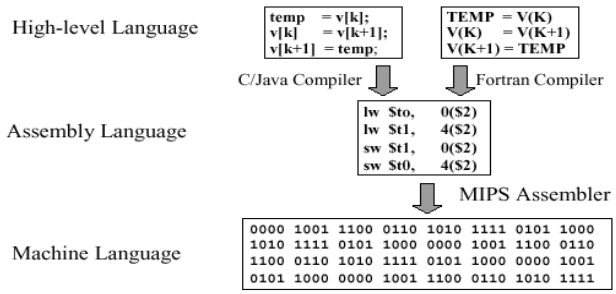

Fig. 24: el proceso de traducir un programa a código de máquina

La potencia computacional no es el único elemento que define la rapidez o precisión con las que la máquina puede interpretar los signos binarios. También la forma en la que un programador escribe su programa juega un papel importante. Los informáticos utilizan la noción de *cota superior asintótica*, en programación, para clasificar los algoritmos en función de su uso del espacio y el tiempo (la memoria y el procesador). Un mal programador puede hacer que una computadora muy potente quede infinitamente confusa, por ejemplo, cuando el programa tiene un bucle infinito.

Desde la perspectiva de una teoría general de los signos, un lenguaje de máquina consta únicamente de instrucciones de cuatro tipos: aritmética, lógica, transferencia de datos y ramas (también llamado control de flujo)[7]:

- Aritmética: add (suma), sub (resta), mult (multiplica), div (divide)
- Lógico: and, or, srl (desplaza lógicamente a la derecha), ssl (desplaza lógicamente a la izquierda)
- Transferencia de datos: lw (carga la palabra), sw (almacena la palabra), lui (carga lo superior inmediato)
- Sucursales:
    - Condicional: beq (ramifica si igual), bne (ramifica si no es igual), slt (establece en menor que),
    - Incondicional: j (salta), jr (registra el salto), jal (salta y haz enlace)

Claramente este es un lenguaje diseñado para recibir y ejecutar órdenes, sus verbos solo son imperativos que se refieren a espacios físicos en la memoria o dispositivos de entrada y salida. Esto prueba que una máquina no posee voluntad. No obstante, aquí es posible adoptar un enfoque lacaniano de la dialéctica del amo y el esclavo entre el usuario y la máquina, y hablar del deseo y su papel en la formación de la subjetividad del usuario. Haciendo una analogía psicoanalítica, de acuerdo con Lacan, la máquina:

[7]https://www.cise.ufl.edu/~mssz/CompOrg/CDA-lang.html

> [does not come] to control the master and moves beyond his designation in the binary master/slave. As the producer of the master's objects of desire, the slave gradually comes to govern the satisfaction or suspension of the master's desire, and thus to control the master's desire in a roundabout way.[8]

Sin embargo, siendo la máquina un artefacto cultural, es la estructura cultural en la que la máquina se produce la que toma el control sobre el maestro/usuario a través del mecanismo explicado por Lacan y juega un papel importante en la formación de subjetividad para el usuario. Aquí, cuando me refiero a la subjetividad de una máquina, considero a una máquina como parte de un sistema mayor que tiende una red de subjetividades entrelazadas; así, el objeto máquina opera como una interfaz para esa subjetividad.

Fernando González Rey explica:

> Culture is a symbolic system within which various human practices and normative systems foster life for the persons who share one particular culture. Symbolic realities are human productions that embody different histories that objectify themselves in language, discursive practices, social representations, myths, normative systems, religions and other cultural productions. Within these symbolic networks of cultural facts, individuals, institutions and groups develop their complex and singular subjective organizations, which represent truly subjective productions. In these productions, emotions are embedded within symbolic processes, which turn subjectivity into an intrinsic component of culture. [Ray, 2017, p. 182]

Como hemos dicho antes, al contrario que Saussure, Peirce se niega a eliminar al sujeto actuante y al cogito cartesiano en su semiótica. Colapietro explica que frente a la tradición mentalista en la que los signos son expresiones de la mente, Peirce considera la mente como una especie de semiótica. Es decir, la realidad de la mente se considera esencialmente el desarrollo de un sistema de signos. Aunque se apoya en Peirce, Eco intenta eliminar al usuario concreto de los signos de su teoría

[8] *Encyclopedia of Psychoanalysis*, https://nosubject.com/Master/Slave_Dialectic

general. Una computadora es un usuario de signos y encaja perfectamente en la noción de semiótica de Eco, que es una teoría de códigos más una teoría de la producción de signos. Sin embargo, ni Colapierto ni Peirce eliminan el tema de la semiótica.

La máquina es una interfaz con la subjetividad de la cultura que la produce; sin embargo, cuando nos referimos a las computadoras, no podemos eliminar los aspectos objetivos de cada computadora: por ejemplo, a nivel físico, una computadora de 32 bits puede acceder a 2^{32} direcciones de memoria diferentes y una computadora de 64 bits a 2^{64} direcciones de memoria. En un nivel funcional, un programa (un mensaje codificado) no se puede compilar (interpretar) en el procesador ARM, pero se puede interpretar en una computadora basada en Intel. En palabras más simples, una computadora que sólo puede ejecutar un sistema operativo de hace veinte años no puede leer/interpretar/ejecutar la mayoría de los programas/mensajes dirigidos a nuestras computadoras recientes. En teoría, todas esas computadoras usan señales 0 y 1 y sus sistemas de signos siguen patrones similares, sin embargo, actúan de manera diferente. En realidad, existen varios lenguajes de máquina que son diferentes de una arquitectura de máquina a otra. ARM, DEC, x86, IBM, MIPS son algunos ejemplos de lenguajes de máquina. Son específicos de una familia de procesadores y un código escrito para un procesador no se puede ejecutar en otro procesador de otra familia.

III-II

La subjetividad de un software

Hemos mencionado en capítulos anteriores que la materialidad de un poema digital es la de un programa o software. Aunque en su forma pura está hecho de series de 0 y 1 que solo son interpretables por la máquina, un software tiene paralelamente otra forma comprensible para la noción humana de un programador. Así, el software es un artefacto cultural que existe para traducir y operar entre dos subjetividades. El texto en el que se crea no es el texto en el que "vivirá". Se crea mediante un texto en un lenguaje de programación y "vivirá" como un texto en código máquina. El primer texto es la idea y el segundo, el cuerpo. Para cambiar el cuerpo, un programador debería cambiar la idea. La máquina es un ser socrático puro.

El sistema de signos en el que opera un programa se denomina lenguaje de programación. Wikipedia menciona que hay 700 lenguajes de programación y la *Online Historical Encyclopaedia of Programming Languages* (https://hopl.info/home.prx) cuenta con el registro de 8,945 lenguajes de programación desde el siglo XVIII hasta el presente. Se puede crear un software utilizando uno o varios idiomas al mismo tiempo. Incluso podemos hablar de las subjetividades de un software. Un lenguaje de programación se puede clasificar de varias maneras. Por ejemplo, puede ser endógeno o exógeno; puede ser algorítmico, funcional, estructural, reflexivo, conversacional, imperativo, espacial, orientado a la

operación, orientado a la expresión o incluso fenomenológico. Todas estas categorías pertenecen a una dimensión cognitiva diferente de la cultura. También hay lenguajes que están diseñados para ser multiparadigmáticos. Dos software que tienen una funcionalidad exactamente similar pueden originarse a partir de dos categorías cognitivas opuestas o sistemas de signos diferentes. En el estado de la existencia de un programa, es decir, cuando se está ejecutando, no se puede determinar fácilmente el sistema de signos en el que fue creado. Cualquier programa tiene un código fuente que se compone de los siguientes elementos:

1. Metadatos, encabezados y variables ambientales.
2. Cuerpo del código que consta de definiciones e instrucciones.
3. Comentarios y documentación.
4. Estructuras de datos internas o externas.

Además, cualquier programa tiene un control de versiones que define las etapas de su proceso de desarrollo. Los códigos de programa están sujetos a derechos de autor y están escritos de acuerdo con una licencia.

En los siguientes apartados primero analizaremos la plataforma en la que ELO publicó los trabajos compilados en sus antologías. Después analizaremos algunas las obras seleccionadas de poesía generativa, Twitter e hipertexto publicadas por ELO desde la perspectiva de su código fuente. Hay dimensiones de investigación

relacionadas con el análisis del código fuente en las que es necesaria una formación en informática. Para los efectos de esta tesis, evitaré profundizar en ellos.

III-II-I

Plataforma de publicación de software

La página en la que se publican los poemas digitales de las antologías ELO actúa como una plataforma de intermediación digital, es decir, intermedia entre el trabajo y la comunidad o los visitantes. Las antologías sirven también como repositorios de las obras seleccionadas. En la primera y segunda antologías (publicadas en 2006 y 2011), la plataforma sirvió también como anfitrión de varios trabajos publicados. En ese momento, la mayoría de los trabajos eran hipertextuales o basados en Adobe Flash.

El servidor web ELO podía fácilmente actuar como un anfitrión porque no requería una configuración especial o complicados servicios de base de datos, aplicaciones o seguridad; por ello, el autohospedaje era una opción. Para algunos trabajos diferentes existía un enlace de descarga e instrucciones de instalación. Por ejemplo, *Galatea*, de Emily Short, viene con estas instrucciones:

> Download and install Spatterlight if you do not already have a z-machine interpreter. Download and unzip Galatea.zip and open the resulting file Galatea.z8 in your interpreter.

El enfoque de la tercera antología (publicada en 2016) cambió hacia una presentación más detallada del trabajo y la preservación del código fuente. En casos como *Uncle Roger* de Judy Malloy, el trabajo también se aloja en su servidor.

Al respecto de las comunidades de desarrollo de software, se puede mencionar que SourceForge comenzó en el año 1999. Ya para el año 2006 era una plataforma bastante famosa para compartir el código fuente de los programas de código abierto. Al mismo tiempo proporcionaba un sistema de control de revisión, wiki, métricas y análisis, acceso a la base de datos y un subdominio único para presentar cada programa. En septiembre de 2020, SourceForge afirmó albergar más de 502,000 proyectos y tenía más de 3.7 millones de usuarios registrados.

Github es otra plataforma importante para los desarrolladores de software. Comenzó en 2007 y para 2011 era una de las plataformas más conocidas para compartir código. GitHub ofrece el control de versiones distribuidas y la funcionalidad de administración de código fuente. Proporciona control de acceso y varias funciones de colaboración, como seguimiento de errores, solicitudes de funciones, gestión de tareas, integración continua y wikis para cada proyecto.

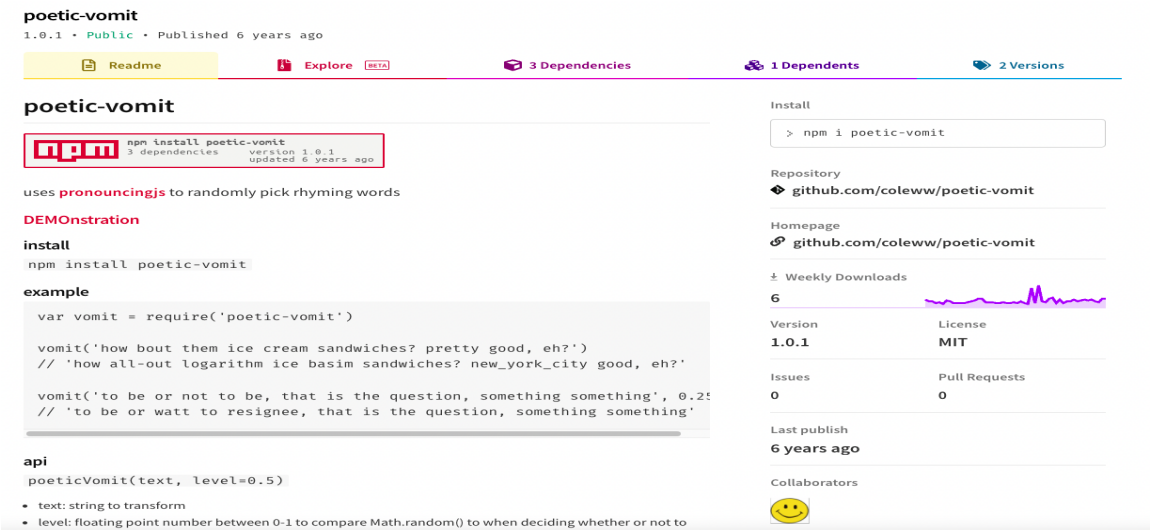

Fig. 25: presentación de una obra digital en NPM

En noviembre de 2021, GitHub informa tener más de 73 millones de desarrolladores y más de 200 millones de repositorios. También hay otras plataformas como npm que brindan servicios similares para la comunidad de desarrollo de software.

Para comprender el trasfondo ideológico de cada tipo de publicación, será útil comparar un poema generativo, publicado tanto en un repositorio de desarrollo de software como en el sitio web de ELO.

@everyword de Allison Parrish se presenta en Github como un script simple para crear servicios de Twitter. La descripción es una oración y el texto que respalda la descripción del proyecto viene con instrucciones técnicas sobre la versión del lenguaje de programación, las dependencias, la estructura de datos, los métodos de ejecución, las notas de producción, la configuración de la plataforma Twitter y la licencia (en este caso, la licencia MIT). Además de un explorador simple que nos brinda la posibilidad de navegar en el código fuente y leer los códigos, la plataforma

también brinda información sobre cómo, desde su inicio en 2015 hasta ahora, el proyecto interactuó con la comunidad de desarrolladores como contribuyentes, o la frecuencia con la que el poeta/programador cambió su código, o cuántas personas lo han utilizado en sus proyectos. En la sección denominada solicitudes de extracción, los comentarios de otros desarrolladores son visibles y se puede ver cómo contribuyeron a mejorar el trabajo y resolver errores o conseguir un mejor rendimiento.

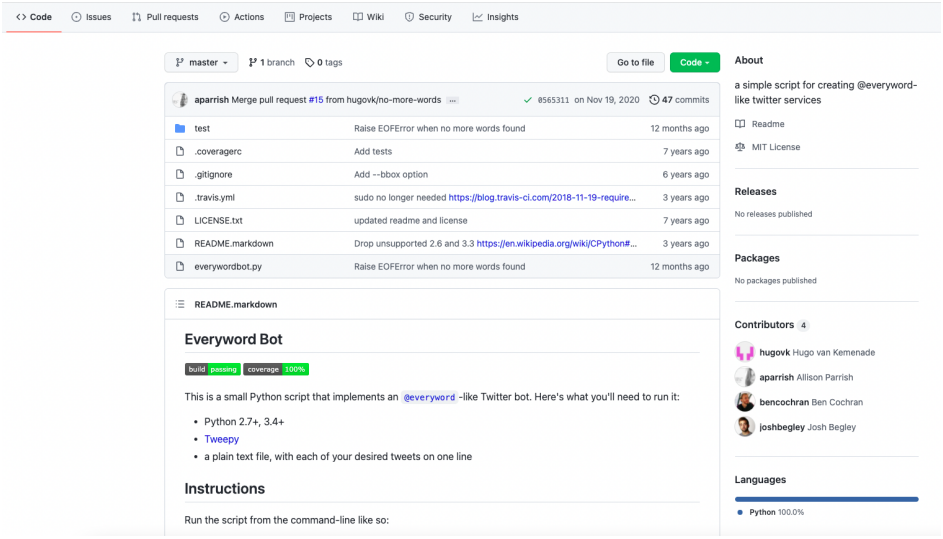

Fig. 26: presentación de una obra digital en GitHub

El mismo trabajo ubicado en la plataforma ELO tiene un párrafo de declaración de la artista, un párrafo como biografía, un párrafo de declaración editorial, una sección de metadatos que proporciona información general como palabras clave, el año de publicación y un archivo y código fuente descargables. Esta página sirve principalmente como catálogo. Aunque se puede acceder al código

fuente, está envuelto en un archivo zip y no es visible hasta que se descarga. Además, no hay rastro del proceso de desarrollo, la documentación o la licencia del código fuente. Finalmente, la página de ELO no proporciona información sobre el repositorio de GitHub del trabajo.

Fig. 27: presentación de una obra digital en la página de ELO

La plataforma Github es interactiva, permite comentar, colaborar, compartir, etc. El catálogo de ELO no lo es. Ni siquiera hay una sección de comentarios en la plataforma. La plataforma ELO sirve como una antología en su sentido tradicional, que es una colección de obras literarias elegidas por un compilador. Incluso evita la interactividad, que es una posición ideológica de la teoría del hipertexto.

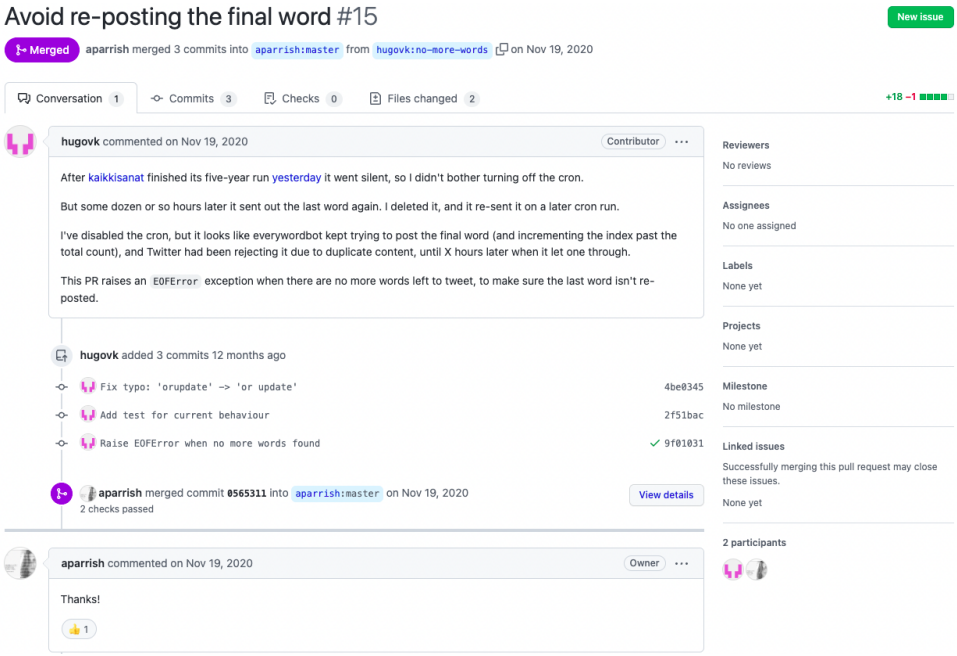

Fig. 28: comentarios sobre Everyword Bot en GitHub

La comparación que aquí se presenta nos da una idea de algunas contradicciones ideológicas que es importante tener en cuenta cuándo analizamos las obras. ELO y el trabajo presentado en ELO (en el ejemplo mencionado, la obra que incluye los detalles expresados en GitHub) son dos entidades diferentes y, a veces, opuestas.

En *Amateurs Online*, Yra Van Dijk dice que la creatividad en una comunidad arroja luz sobre esta paradoja ideológica. Concluye el ensayo enfatizando que: “genres no longer need an institutional base in the offline world: “they exist without bookstores, libraries or the press.” Es decir que existe una identidad comunal en la que cuestiones formales y técnicas sobre el género y su software se abordarían en la

comunidad, y “the creativity of the community is to be found in the exchanges themselves.” [Van Dijk, 2015, p. 66]. Más adelante, agrega:

> The interactive fiction community is well documented and has highly structured archives of both texts and paratexts. These well archived exchanges function as a reservoir of critical, technical and poetical knowledge and theory. This implies a form of collaboration with a “product,” which is however not to be measured in any pragmatical or economical sense of the word, but rather as a form or resistance against that order (cf. Nieckarz 2005, 421). In addition to this reservoir of exchanges, the individual members of the community of IF-developers produce works of IF, which are—to state again—not to be measured in symbolic or economic value, but in pleasure and creativity on the part of the producer and the consumers... the exchanges themselves perform institutional functions: criticism, canonization, the writing of the “history” of the genre and its influences, its distribution, etc.[Van Dijk, 2015, p. 67].

III-II-II

Metadatos, encabezados y variables ambientales

En *Examining Paratextual Theory and its Applications in Digital Culture*, Patricia Tomaszek y Nadine Desrochers analizan *The Unknown: The Original Great American Hypertext Novel* de William Gillespie, Scott Rettberg, Dirk Stratton y Frank Marquardt. Utilizan un enfoque paratextual para la lectura de metadatos y comentarios en una obra de literatura digital.

Para Genette el concepto de paratexto consiste en una serie de elementos que pueden ser portadas, tabla de contenido, prefacios, notas, etc. que acompañan a una narración para formar un libro y presentarlo al lector. El estudio de Tomaszek y Desrochers utiliza un enfoque interdisciplinario para responder las siguientes preguntas:

1) ¿Cómo apoya el paratexto de *The Unknown* la experiencia lectora y el descubrimiento de información sobre la génesis y organización de la obra?

2) ¿Cómo se establece u obstaculiza la confiabilidad en el paratexto de *The Unknown* y qué efectos tiene esto?

3) ¿Es la teoría paratextual una herramienta pertinente para el estudio interdisciplinario de una pieza de literatura electrónica como *The Unknown*? [Desrochers. 2014, p. 165]

Las autoras de esta reflexión también utilizaron dos navegadores web, Safari y Mozilla Firefox, que les mostraron algunas diferencias en los datos, por ejemplo, en la visualización de las páginas de inicio y del mapa. Usaron el código fuente html que les proporcionó uno de los autores. Cabe mencionar que utilizaron como método de análisis un enfoque hipertextual y una lectura cercana. También realizaron un análisis de contenido cualitativo como segundo método de análisis. No fue una tarea fácil aplicar la teoría de Genette en este tipo de textos. Como lo mencionan en la sección "Hallazgos del ensayo", encontraron indicaciones contradictorias sobre el comienzo y el final de la obra:

> The situation concerning the work's end is similar. Given that there is no table of contents on the home page, there is also no indication of where the work ends. In the table of contents of the red line, however, there is a title that links to "The End" (theend.htm). [Desrochers. 2014, p. 167]

También hay contradicciones cuando hacen énfasis en las fechas en las correspondencias:

> Some of the dates provided in the orange line differ from one titular unit to the other, with differences ranging from one day to one year. Such inconsistencies are brought to light by comparing the date listed in the index with either the date provided in the title tag of the page itself (when available) or with the file name contained in the URL, which in seven cases is constructed from numbers resembling an indexed date. [Desrochers. 2014, p. 170]

Y concluyen afirmando:

> It cannot be established whether these quirks and inconsistencies were deliberate, as elements of the ever-expanding maze created by the work's hypertextuality, or are

> simply the result of a relaxed attitude towards the use and presentation of paratextual elements and the organization of the work in general. [Desrochers. 2014, p. 173]

Después de un análisis cuidadoso del código fuente y la organización de los mismos dentro del trabajo, las autoras llegaron a aplicar la teoría de Genette para comprender dónde está la desconexión de la intención original de crear "una infraestructura paratextual fuerte, capaz, a su vez, de apoyar la exploración y el estudio de la organización del trabajo". De esta manera, llegan incluso a mencionar que:

> The "defamiliarized" paratext of *The Unknown* certainly makes it difficult to capture the "whole" of the piece from a structural or information sharing point of view, as it constantly defeats attempts to index, label, or classify its content. [Desrochers. 2014, p. 180]

La conclusión de su estudio es aún más impactante desde el punto de vista de un programador:

> Combined qualitative content analyses and close readings reveal a plethora of inconsistencies in the titular apparatus and other paratextual elements whose role should be to provide identifiers and references for the reader. Furthermore, comments and notes appearing in the source code seem to be both informational and misleading, as testimonies and authorship claims collide. These inconsistencies affect the study of the work's structure, its presentation as a collection of HTML pages, the researcher's understanding of the work's genesis, any indexing efforts, the narrative itself, and, of course, the reading experience it provides. [Desrochers. 2014, p. 181]

Un análisis paratextual de *The Unknown* nos arroja inconsistencias e información engañosa dentro de la propia obra. El problema no es la metodología utilizada por las investigadoras mencionadas. Como hemos comentado en capítulos anteriores, el enfoque de la creatividad de Csikszentmihalyi sugiere que la creatividad sea analizada en tres dimensiones o entidades: los individuos, el campo y la cultura. Según los autores individuales de *The Unknown*, este trabajo es lo

suficientemente creativo y está listo para ser lanzado. Sin embargo, el campo en este caso no puede limitarse a la Comunidad de Organización de Literatura Electrónica. La obra publicada es una pieza literaria y un software al mismo tiempo. La comunidad de Electronic Literature Organization aceptó el trabajo para su publicación en la segunda antología, por lo que el trabajo cumplió con sus importantes criterios. Sin embargo, considerando la dimensión *soft* del trabajo, hay varias preguntas sin resolver relacionadas con los criterios de lanzamiento de software que puede arrojar luz sobre la brecha entre las dos dimensiones del mismo campo creativo.

Por ejemplo, todas las obras de las antologías ELO se publican bajo una licencia Creative Commons. Esta licencia, aunque es una licencia común dentro de las comunidades de arte creativo, no es adecuada para el software. El sitio web oficial de Creative Common License explica:

> Unlike software-specific licenses, CC licenses do not contain specific terms about the distribution of source code, which is often important to ensuring the free reuse and modifiability of software. Many software licenses also address patent rights, which are important to software but may not be applicable to other copyrightable works. Additionally, our licenses are currently not compatible with the major software licenses, so it would be difficult to integrate CC-licensed work with other free software. Existing software licenses were designed specifically for use with software and offer a similar set of rights to the Creative Commons licenses.[9]

Este tipo de conflicto de licencias es una evidencia de la brecha mencionada: en Github, *@everyword* de Allison Parrish aparece con la licencia MIT, y en la antología ELO, bajo Creative Commons.

[9]https://support.skillscommons.org/faqs/category/8creative-commons-requirement/#:~:text=Unlike%20software%2Dspecific%20licenses%2C%20CC,reuse%20and%20modifiability%20of%20software.

Independientemente de la metodología de desarrollo que se utilice para escribir un software, existen condiciones que deben cumplirse para poder lanzarlo. Menciono algunas de ellos aquí como los criterios de prelanzamiento:

1) Legal: riesgos asociados, licencias, leyes y regulaciones (por ejemplo, exportación, seguridad).

2) Servicio/Capacitación: el impacto de los cambios, actualización del material de capacitación.

3) Soporte técnico: un documento de funcionamiento, el impacto de los cambios en el funcionamiento, documento de solución de problemas.

4) Experiencia del usuario: documentación en línea o impresa del software.

5) Gestión de la configuración: siguiendo las prácticas de control de cambios.

6) Desarrollo: defectos solucionados, documentación de desarrollo, pruebas unitarias / de integración.

7) Gestión de productos: seguimiento de la resolución previa de defectos.

Esta lista puede ser más detallada según la cultura del equipo de desarrollo de software. Un análisis paratextual de los trabajos presentados en las tres antologías sería posible si la plataforma siguiera los criterios de lanzamiento de software del campo. Para dar ejemplos sin profundizar en la disciplina de la informática, y en este caso sobre el tema de “Desarrollo en la lista de verificación de la versión de software

(n. ° 6)”, menciono la sección de comentarios de *Blue Hyacinth Stir Fry Text* de Jim Andrews y Pauline Masurel.

```
<!--
Thanks to Pauline Masurel for a great text and the blue squares and
the text colors.
Thanks also to Marko Niemi for showing me how to fix the code
so it runs on many more browsers than the original version ran on.
I wrote the original code in 1999 and it ran only on IE for the PC. Marko
showed me in 2004 how to get it running on the Mac and PC on IE 4+,
Netscape 7+, Firefox, and Safari. Haven't tested it on Opera.
Jim Andrews
November, 2004
-->
<html>
<head>
<title>Blue Hyacinth Stir Fry Text</title>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-
1">
<META NAME="description" CONTENT="Stir Fry by Pauline Masurel">
<META NAME="keywords" CONTENT="Stir Fry, Pauline Masurel">
<META NAME="AUTHOR" CONTENT="Jim Andrews, Pauline Masurel">
```

El código fuente del programa comienza con una sección de comentarios antes de las etiquetas <html>. De acuerdo con el estándar HTML 4.01 que se introdujo en el año

2000, se sugiere que el programador use <! DOCTYPE html> antes de <html> y que los comentarios vengan después de la definición de <DOCTYPE>. Cualquier comentario anterior forzaría al navegador a leer el código fuente en modo peculiar, lo cual no se recomienda y, como resultado, puede causar un comportamiento inesperado en el navegador.

El estándar es claro al respecto:

> There are now three modes used by the layout engines in web browsers: quirks mode, almost standards mode, and full standards mode. In quirks mode, layout emulates nonstandard behavior in Navigator 4 and Internet Explorer 5. This is essential in order to support websites that were built before the widespread adoption of web standards. [...] For HTML documents, browsers use a DOCTYPE in the beginning of the document to decide whether to handle it in quirks mode or standards mode. To ensure that your page uses full standards mode, make sure that your page has a DOCTYPE. [10]

Como explica Wikipedia:

> En modo estándar las páginas se producen de acuerdo con las especificaciones de HTML y CSS, mientras que en el modo "quirks" tratan de emular el comportamiento de los navegadores viejos para asegurar que dichas páginas se representen de acuerdo a la intención original de sus autores.
>
> Algunos navegadores (aquellos basados en el motor de dibujado o renderizado Mozilla Gecko, por ejemplo), usan también un modo "casi estándar" que intenta mantenerse entre los dos, emulando algunos fallos viejos mientras la mayoría lo hace conforme a las especificaciones. [11]

[10]https://developer.mozilla.org/en-US/docs/Web/HTML/Quirks_Mode_and_Standards_Mode
[11]https://es.wikipedia.org/wiki/Quirks_Mode

En este caso, aunque el código fuente no mostrará ningún error, la recomendación del campo es especificar dónde, cuándo y cómo un programador decide usar un modo peculiar (para códigos fuente más antiguos) y la solución está en escribir un código que tenga una estructura como esta:

<!--[if anybrowser]> this <![endif]-->

O justo después de <html>

!--[if IE]><meta http-equiv="X-UA-Compatible" content="IE=8" ><![endif]-->

Por ejemplo, *Argot Ogre, OK!* de Andrew Plotkin es consistente con este estándar y por eso comienza su largo comentario después de la etiqueta <html>.

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html manifest="argot-ogre-ok.manifest">

<head>

[Algunos códigos]

<!--

Argot Ogre, OK!

(Andrew Plotkin, Sept 27, 2011)

http://eblong.com/zarf/argot-ogre-ok.html

This code is public domain. Text is by the authors attributed below.

Renderings: Contemporary Japanese Poetry Generator de Shinonome Nodoka es otra obra presentada en el tercer volumen de las antologías ELO. Aparte de la falta de modularidad en el código fuente, solo se puede acceder a partes de la información paratextual sobre el proyecto a través del código fuente. En la definición de la hoja de estilo del trabajo, vemos una línea como esta:

```
a[href*=notes] { display: none; }
```

Esta línea obligará al navegador a no mostrar las notas (a las que se puede acceder aquí: https://s3.amazonaws.com/curamag/renderings/notes.html#contemporary). Pero si a través del código fuente encontramos esta información paratextual y visitamos el sitio web, hallamos un relato interesante de los proyectos relacionados y las inspiraciones del poeta / programador. ¿Por qué tal información debería ocultarse al lector? Y si no es una información útil, ¿por qué no se elimina del código fuente? En un enfoque estándar, si el programador desea incluir dicha información como comentario, la incluirá en la sección de comentarios. No hay una razón clara por la cual el programador la incluya en el código en ejecución y luego la oculte mediante la instrucción incluida en la hoja de estilo. Este es un problema relacionado con la *gestión de la configuración* en la lista de criterios de prelanzamiento de software (nº 5).

No todas las obras de la antología se ajustan a los criterios de prelanzamiento. Por ejemplo, The Last Performance de Judd Morrissey viene con un plano de

documentación y los trabajos independientes siguen las reglas de las comunidades informáticas para publicar sus trabajos. PoetRNN de Sam Ballas ofrece una descripción bastante completa de la lógica del programa. Las funciones van acompañadas de información paratextual como:

> Static methods in this class seemed to make testing easier. This class is largely taken from Andrej Karpathy.

Sin embargo, en la plataforma Github de *PoetRNN*, hay dos problemas abiertos que abordan errores en la administración de la memoria. El autor parece haber perdido interés en el proyecto y no hay respuesta ni reacción a las preguntas desde 2015 hasta ahora.

En cambio, *Poemas: RNN* de Denis Krivitski está muy bien documentado. El autor primero publica un ensayo sobre la lógica detrás del programa y explica la arquitectura. El código está bien estructurado y las ambigüedades se aclaran mediante comentarios.

Debido a que esta tesis forma parte de la disciplina de las humanidades, profundizar más en los códigos me obligaría a usar terminologías complicadas demasiado arraigadas en la ingeniería informática. Por ello, trato de evitar hablar de cuestiones más técnicas sobre los códigos fuente. Confío en que con estos pocos ejemplos, se puede entender cómo la falta de un estándar comunitario en el lanzamiento de un software causa inconsistencias y ambigüedades en la información paratextual de los códigos fuente.

III-II-III

El cuerpo del código fuente, definiciones e instrucciones

Hay varias escuelas de pensamiento dedicadas a la comprensión de los códigos fuente. Tener un código limpio puede reducir el costo del desarrollo futuro de un software, ayudar a los contribuyentes externos o futuros a mejorar o actualizar el programa y reducir o evitar errores. La lista de los beneficios de un código limpio puede seguir y seguir.

Pero, ¿qué es un código limpio? No hay una respuesta sencilla a esta pregunta. Cada escuela de desarrollo de software tiene su propio estándar. Sin embargo, existen puntos en común entre todas las escuelas. Hay valores, métricas y conocimientos que nos ayudan a reconocer un código limpio de uno sucio. Utilizo las sugerencias de Robert C. Martin sobre código limpio. Su noción de este concepto está bien establecida en las comunidades de desarrollo de software. Martin, como lo describe Wikipedia, es más reconocido por desarrollar muchos principios de diseño de software y por ser uno de los fundadores del influyente *Manifesto for Agile Software Development* (https://agilemanifesto.org).

Para definir una noción de código limpio, Martin intenta introducir primero este concepto a través de la visión de otros profesionales. Bjarne Stroustrup, inventor de C ++, por ejemplo, define un código limpio como agradable y elegante en apariencia o forma; agradablemente ingenioso y sencillo. Grady Booch, autor de

Object-Oriented Analysis and Design with Applications lo describe de la siguiente manera:

> Clean code is simple and direct. Clean code reads like well-written prose. Clean code never obscures the designer's intent but rather is full of crisp abstractions and straightforward lines of control. [Martin, 2009, p. 8]

Para Dave Thomas, el fundador de Object Technology International, la limpieza de un código está en la legibilidad y la posibilidad de ser mejorado por otros desarrolladores. [Martin, 2009, p. 9] Además, Ron Jeffries sugiere evitar la duplicación y la expresividad para llegar a un código limpio. [Martin, 2009, p. 10]

La mayoría de estas expresiones se parecen a las recomendaciones de expertos literarios para redactar un buen ensayo literario o una prosa específica. Quiero decir, ambos campos de la escritura, ya sea el código fuente o la literatura, tienen sus propios estándares para calificar un texto. Un código fuente no es solo una herramienta o unas instrucciones para integrar un programa, sino que dentro de la comunidad informática se valora como un texto mediante sus propias calificaciones literarias y científicas.

Es necesario mencionar algunos de los estándares de calificación de un código en relación con la literatura digital:

El primer capítulo del libro de Martin sobre cómo escribir un código limpio explica cómo nombrar códigos. Un nombre debe tener significado en un programa, es decir, debe revelar la intención del programador, evitar la desinformación, hacer distinciones significativas y debe ser pronunciable. Además, es mejor que se le pueda

hallar y que el programador tenga que elegir una palabra por cada concepto y así no confundir al lector.

Tiny Star Fields de Katie Rose es un pequeño bot con pocas líneas de código. Genera aleatoriamente un pequeño grupo de estrellas varias veces al día. El programa consta de algunas variables: “randFin”, “starArray” y “stars”. Aquí, el nombre “randFin” no tiene ningún significado obvio. Podemos interpretar el “rand” como aleatorio, pero ¿cuántas variaciones podemos imaginar para “Fin”?

El atributo ID especifica una identificación única para un elemento o etiqueta en HTML. En su definición de ID de las etiquetas, *Lexia to Perplexia* de Talan Memmott no sigue ningún estándar. Un ID, sin ningún motivo puede aparecer en mayúsculas o no mayúsculas. La denominación de los ID no tiene un significado obvio: sremoteb, sremoteb, tlocal, slocal, etc.

Pero este no es el caso de la mayoría de las obras. Los códigos fuente de la poesía de Twitter están muy alineados con las convenciones de la nomenclatura. Probablemente, debido a la naturaleza de su arte creativo que está incrustado en las redes sociales, siguen las instrucciones de la comunidad informática más que otros géneros de la literatura digital.

Por ejemplo, *Two Headlines* de Darius Kazemi utiliza la convención de casos de camello (camelCase) y su denominación es muy agradable y reveladora. No es difícil captar el significado de una línea como esta:

```
function findCommonText (headlines)
```

Esta función busca textos comunes dentro de los titulares. Wikipedia define la convención camelCase como la práctica de escribir frases sin espacios ni puntuación, lo que indica la separación de palabras con una sola letra en mayúscula y la primera palabra que comienza con mayúscula o minúscula.

Aunque algunos artistas están siguiendo las convenciones de nomenclatura de su comunidad en el campo de la programación y otros evitan o desconocen las convenciones, la falta de un estándar o un acuerdo sobre estos temas es evidente. Esto prueba la identidad dispersa de los artistas en el campo de la literatura digital. También podría interpretarse como una diversidad. Si tal diversidad se conduce a un esfuerzo colectivo de redacción de estándares o convenciones se materializará una comunidad.

Aparte de la importancia del nombre, Robert C. Martin se centra en otras dimensiones del código fuente. Las funciones que son modulares en cajas de instrucciones deben ser pequeñas, tienen que hacer una cosa y no varias cosas al mismo tiempo, deben tener un nivel de abstracción, deben usar nombres descriptivos. Además, un programador no debería repetir sus designaciones anteriores y debería intentar definir funciones utilizables en el futuro, etc. La misma identidad dispersa/diversa puede observarse en esta dimensión de los códigos fuente. El comentar y formatear el código fuente también tiene sus propias convenciones. [Martin, 2009, pp. 17-52]

Conceptos como el tipo abstracto de datos (TAD), el manejo de errores, las pruebas unitarias y muchos otros aspectos de la programación son muy importantes para ser considerados aquí, sin embargo, deberían ser temas de algunas otras tesis en el campo.

III-II-IV

Comentarios y documentación.

Los comentarios en los códigos fuente actúan como las notas a pie de página en un artículo. Martin cree que son evidencias de las fallas de un programador al expresarse por medio del código. Por lo tanto, si se incluyen en el código, deben tener un propósito y ser mínimos. Si un código no es legible, los comentarios no pueden subsanar su problema. Es una buena práctica hacer comentarios sobre los aspectos legales del código, brindar alguna información básica, una explicación de la intención del código, hacer algunas aclaraciones, advertir sobre las consecuencias de un mal uso de las instrucciones, dar un plan de desarrollo futuro mediante una lista de TODO (asuntos por hacer), amplificar la importancia de algún aspecto particular del código, etc.

A diferencia de los comentarios que aparecen dentro del código, la documentación actúa principalmente como un manual para leer el código fuente y su arquitectura y, en ocasiones, es incluso una guía aparte junto al código. La documentación es una parte necesaria de cada artefacto o software digital. La documentación nos lleva a través del código, mostrándonos su arquitectura, dependencias, contextos, casos de uso, hoja de ruta de desarrollo, problemas, plan de características, etc. Sin una documentación adecuada, la conexión entre el software y su desarrollo futuro se perderá. El código perderá la capacidad de encarnación dentro

de la cultura del campo. Las comunidades de desarrolladores de software rara vez confían en un software que no proporciona documentación. En ocasiones, los trabajos incluidos en las antologías de ELO vienen con una nota, un manual de instrucciones, breves comentarios de explicación dentro del código, pero no con una documentación. Parte de la información paratextual mas reveladora sobre un código se puede encontrar en su documentación. Por ejemplo, los cambios en las versiones del software, o licencias, abreviaturas utilizadas dentro del código, etc.

La falta de documentación en las antologías de ELO solo es otra evidencia de la gran brecha entre estas y el campo de la computación. Cualquier comparación con otro movimiento artístico en el mundo digital, como el desarrollo de juegos, muestra las diferencias en su uso de la documentación. Los videojuegos como *0 A.D.*, *AstroMenace* o *Cataclysm: Dark Days Ahead* son algunos ejemplos. Este último incluso está documentado en una wiki completa. Como mencioné en los capítulos anteriores, los juegos de computadora y la poesía digital comenzaron al mismo tiempo. Sin embargo, el juego tomó el camino de las comunidades de software y la industria y, como Markku Eskelinen señala, la literatura digital permaneció vinculada a los departamentos de humanidades durante mucho tiempo.

III-II-V

Estructuras de datos internas o externas.

Las estructuras de datos y las bases de datos son otro aspecto importante de una base de código. Son un campo de la informática en sí mismo. A continuación doy algunos ejemplos para mostrar problemas parciales de la mencionada desconexión o brecha entre las comunidades Digilit y las de desarrollo de software. Los desarrolladores de software hacen todo lo posible para escribir un código que sea lo más modular posible. Intentan desarrollar una lógica y una ontología para el programa, mantener una arquitectura mínima, reutilizable, segura y escalable. Por ejemplo, separan la capa de datos de la capa de aplicación en un programa. Al revisar *Frequency* de Scott Rettberg, un desarrollador de software se pregunta por qué el programador almacenó los datos del programa en un formato de archivo html. Hay varias soluciones organizadas disponibles, incluso como una base de datos estática, como archivos json, etc. Los archivos se denominan “Frequence” más un número. El nombre no indica nada sobre el contenido. Cada archivo, al ser un documento html, está acompañado de etiquetas repetidas que se pueden programar para que se generen automáticamente. No hay justificación desde la perspectiva del desarrollo de software para implementar el programa de una manera tan redundante y sin un desarrollo adecuado de la estructura de datos.

En *Renderings: Contemporary Japanese Poetry Generator* de Shinomone Nodoka, los datos y el código se mezclan en un solo archivo, es decir, la aplicación y

la capa de datos no se separan. *Blue Hyacinth Stir Fry Text* de Jim Andrews y Pauline Masurel tiene el mismo problema.

Este no es, por supuesto, el problema de todas y cada una de las obras de la antología. Por ejemplo, *poem.exe*, un bot de Twitter de Liam Cooke, proporciona una organización adecuada del código fuente y los datos. Hay una carpeta dentro del código fuente llamada corpus que contiene un archivo json con el nombre de haiku que indica los datos utilizados por la aplicación.

La estructura en la que se manifiestan los datos dentro de las obras de arte de las antologías de ELO arroja luz sobre las identidades dispersas de estas colecciones. Hay poetas/programadores que tienden a reconocer la noción híbrida del signo en la literatura digital y hay otros para quienes un software es solo una herramienta para materializar sus ideas literarias.

Hay varias otras dimensiones que no abordo en mi análisis, como escalabilidad, optimización algorítmica, marcos arquitectónicos, etc. Espero que esta tesis pueda despertar interés para que otros investigadores se adentren en esas dimensiones del código fuente. Un enfoque comparativo entre el campo de la literatura digital y el desarrollo de juegos con respecto a las dimensiones mencionadas puede ser muy emocionante y fructífero.

Para concluir esta parte reflexionaré sobre *Trauma's of Code*, un artículo de Katherine Hayles perfectamente alineado con la práctica de las antologías de la plataforma ELO. En este artículo, Hayles desarrolla una analogía: así como el inconsciente es para el consciente, el código de la computadora es para el lenguaje

[Hayles, 2013, pp. 40-41]. Incluso expresa que el código es el inconsciente del lenguaje en nuestra cultura computacionalmente intensiva. Para que esa analogía funcione, Hayles oculta el código de la superficie sin negar su poder sobre la superficie misma. Por eso, afirma que una noción ingenua sobre la programación supone que el código es claramente obvio para cualquiera que conozca el lenguaje de codificación. Ciertamente, el código no es claramente obvio para los programadores. Nunca se supuso que fuera así. La cuestión no es que el código sea obvio, sino que sea legible o comprensible. Esa es la razón por la que los programadores llegaron a inventar las convenciones que discutimos en este capítulo. Un código debe ser tan legible como una prosa bien escrita. Entonces, en lugar de argumentar acerca de la transparencia del código, algo que nunca fue el propósito de los programadores, analicemos “trauma del código” asumiendo que la intención del código es la legibilidad. Si no especificamos la noción de un código caeremos en una mistificación del mismo. El código máquina, el código ensamblador y los lenguajes de programación pertenecen a diferentes esferas semióticas de la noción de código y, por lo tanto, si no se especifica su esfera de semiótica, se puede caer fácilmente en la trampa de la dualidad entre el lenguaje natural y el código (como símbolo de una existencia desconocida, ilegible, inaccesible pero poderosa).

Hayles utiliza la noción de programación en equipo como excusa para hablar de la inaccesibilidad de la totalidad del código, pero es obvio que un programa escrito por un equipo también necesita un esfuerzo de lectura colectivo y, por lo

tanto, ningún individuo cuenta con el dominio de la totalidad de la lectura, más bien forma parte de un esfuerzo común. Escribe:

> These examples demonstrate that in practice both code and the unconscious are opaque, although with code it is a matter of degree, whereas the opacity of the unconscious is assumed. [Hayles, 2013, p. 40]

Hayles expande la analogía y encuentra una entre el código y el trauma:

> Experienced consciously but remembered non-linguistically, trauma has structural affinities with code. Like code, it is linked with narrative without itself being narrative. Like code, it is somewhere other than on the linguistic surface, while having power to influence that surface. Like code, it is intimately related to somatic states below the level of consciousness. These similarities suggest that code can become a conduit through which to understand, represent, and intervene in trauma. [Hayles, 2013, p. 43]

Esta forma de analogía, aunque es muy poética y perspicaz contiene cierto grado de antropocentrismo donde todas las analogías se basan únicamente en el comportamiento y la mentalidad humana.

Una de las perspectivas que comparto con Hayles sobre la noción de código es cuando se refiere a Adrian Mackenzie y su maravilloso libro *Cutting Code*. La función que Mackenzie (y Hayles) reconoce para el código - uno de sus puntos principales - puede ser apropiado para el caso que estoy debatiendo aquí: el código como el inconsciente del lenguaje [Hayles, 2013, p. 40]. Sobre el tema de la función del código nos encontramos en un terreno común. Sin embargo, el enfoque que está tomando esta tesis se concentra en la materialidad del código, en una serie de

lenguajes colectivamente creados y usados, en lugar de tratar de interpretarlo como una entidad metafísica, indescifrable, oscura, oculta (o que debe, en nombre de la poesía, ser ocultada).

El enfoque de Hayles sobre el código me recuerda los escritos psicoanalíticos de Carl Jung, donde este psicoanalista ve la tecnología como un modo de conciencia humana y, como afirma Jackson 2Bears, comenzamos a imaginar el lado oculto (latente) de lo tecnológico, lo que siempre permanece oculto bajo lo superficial e inaccesible al pensamiento humano. Sin embargo, cuando comenzamos a asumir un sistema de signos híbrido para la literatura digital, no podemos enfatizar solo una dimensión del signo de manera aislada. En un signo híbrido, ninguna dimensión existe independientemente de las demás. Un signo de cyborg es híbrido y ahora, como predijo Donna Haraway, es nuestro presente, no nuestro futuro, "es nuestra ontología poshumana". En palabras de 2Bears:

> For Haraway, the cyborg becomes a means to subvert the kinds of dualisms that exist in Western traditions, binary opposites (self/other, mind/body, male/female, truth/illusion) that in the past upheld ‘hierarchical dominations’ and in the present enable networks of power under the sign of the ‘informatics of domination.’ [2Bears, 2013, p. 156]

Conclusión:

Hacia una semiótica de la creatividad en la poesía digital

Mi abuela solía decir: el discurso es el viento. Aquí, se refería a la temporalidad del habla en la cultura oral. Al mismo tiempo, esta frase era consciente de la materialidad del discurso y/o de la relación entre el viento y la fonología. En varios países, hay festivales de esculturas de nieve. Uno puede preguntarse por qué un artista puede elegir un material tan efímero para hacer una obra de arte. En los capítulos anteriores, cuando me referí a la necesidad de documentación, un plan de desarrollo, posibilidades de actualización, etc., no estaba tratando de predicar la inmortalidad frente a la temporalidad de los objetos creativos. Durante milenios, las culturas oral y escrita convivieron y tuvieron una interacción mutua entre sí. La cultura escrita no se trata sólo de usar y crear el texto escrito, sino también de la transmisión y transformación de textos escritos y no escritos. Siempre ha existido una relación dialéctica entre el signo oral y el escrito. La materialidad, la subjetividad y la ideología de cada uno eran diferentes. Sin embargo, un signo de cyborg es capaz de transformar esa dialéctica en una hibridación en la que un texto adquiere atributos transitorios y duraderos al mismo tiempo.

En las narrativas folclóricas de muchos países, un cuento tiene varias versiones y, a veces, los folcloristas intentan rastrear la migración de esos cuentos entre naciones y

culturas. Cualquier versión de un cuento folclórico modifica otra versión del mismo cuento. Estas modificaciones ocurren en función de las preferencias de una cultura, una comunidad o un narrador individual. El mismo patrón de control de versiones, transformaciones y migraciones ocurre con el software. Un software migra y sigue transformándose en diferentes versiones de un original, que en ocasiones se pierde u olvida. El mundo digital toma prestadas las prácticas comunitarias de la cultura oral para hacer posibles tales transformaciones. Escribir un programa no siempre es un trabajo para completar una tarea, pues también sucede porque los programadores disfrutan de la experiencia de escribir el código. *Tao of the Programming* de Geoffrey James fue parte de la cultura pop de los programadores a finales de los 80. Este libro forma parte de la literatura para programadores, desmitifica el proceso de codificación y teje una analogía entre la escritura creativa y la codificación. Geoffrey comienza su libro con esta frase:

> Something mysterious is formed, born in the silent void. Waiting alone and unmoving, it is at once still and yet in constant motion. It is the source of all programs. I do not know its name, so I will call it the Tao of Programming.
> If the Tao is great, then the operating system is great.
> If the operating system is great, then the compiler is great.
> If the compiler is great, then the application is great.
> The user is pleased, and there is harmony in the world.
> The Tao of Programming flows far away and returns on the wind of morning. [12]

Y continúa con la definición de los diferentes aspectos de la programación:

> The Tao gave birth to machine language. Machine language gave birth to the assembler.

[12] https://www.mit.edu/~xela/tao.html

> The assembler gave birth to the compiler. Now there are ten thousand languages.
> Each language has its purpose, however humble. Each language expresses the Yin and Yang of software. Each language has its place within the Tao. [13]

El texto habla claramente de límites decrecientes entre el ser humano y la máquina:

> The Grand Master Turing once dreamed that he was a machine. When he awoke, he exclaimed:
> "I don't know whether I am Turing dreaming that I am a machine, or a machine dreaming that I am Turing!" [14]

Las nociones budistas que utiliza Geoffrey James juegan alegremente con el sistema cartesiano occidental.

En el curso de esta tesis, además de incursionar en los límites de la máquina y lo humano en la generación de un objeto creativo, traté de cuestionar la noción cartesiana de profundidad y superficie y, a su vez, traer a la superficie lo que se llamó profundidad. Existe una organización horizontal en la que la máquina, el programa y el lenguaje natural interactúan. Es más un sistema de tuberías que un esquema de jerarquías vertical. Una tubería no transmite trascendencia y solo define el orden lógico / ontológico (en el sentido informático) en el que las entidades reaccionan entre sí. También intenté sugerir un enfoque peirceiano de la semiótica, en el que no se elimina la subjetividad y, por lo tanto, somos capaces de expandir la noción de texto a subjetividades no humanas. Sin tal expansión, una noción de signo híbrido (no binario) no actuará como un híbrido: caerá en el antropocentrismo asumiendo que la semiótica encajará en la subjetividad humana.

[13] ibid
[14] ibid

La investigación para esta tesis se desarrolló de forma modular, ya que es una práctica común en la programación de computadoras, por lo que hay un módulo diseñado para la producción de signos, un módulo para el análisis de la creatividad, un módulo para el análisis de nuevos medios y para la ontología (tanto en su sentido metafísico como informático). Estos módulos forman un todo por medio de una ontología. Es un error ontológico (enfatizo que en su sentido dual) analizar o leer un poema digital en su supuesta superficie. Mi propósito era disminuir la brecha entre el enfoque de las humanidades hacia los objetos creativos digitales y el enfoque de la informática. Por eso, la tesis no se centra mucho en las dimensiones sociopolíticas de tal objeto. Sin embargo, hay indicios y pasajes en los que traté de mencionar los efectos secundarios políticos y culturales de los objetos digitales. El énfasis que hice en el capítulo anterior sobre la importancia de los esfuerzos colaborativos, la transparencia y la naturaleza participativa de las comunidades de programación de software se hizo con tal propósito: abrir una discusión sobre la necesidad de expandir el enfoque editorial tradicional de los objetos creativos digitales. No veo ningún sentido para la existencia de ningún canon o antología cuando se trata de literatura digital. Las atractivas comunidades de artistas creativos en plataformas de desarrollo de software colaborativo, desde los aficionados hasta los profesionales, se mueven a una velocidad tal que escapa a cualquier enfoque canónico. El canon siempre será efímero en un mundo así de vertiginoso y complejo. La comparación que hice entre el desarrollo de juegos y la literatura digital también tuvo este propósito. Si los

departamentos de humanidades conservan la perspectiva de un observador distante de los movimientos digitales y no se involucran en la práctica de sus evoluciones y transformaciones, la brecha entre su mutuo entendimiento aumentará y será la industria la que tomará el control de esta separación y no los artistas individuales o la academia. Este trabajo debe verse como uno de los esfuerzos introductorios, una hoja de ruta, para reducir la mencionada brecha.

De cada capítulo surgen varias preguntas que no pude responder. Primero, por la falta de terminología común, adecuada y entendible por ambas comunidades (literatos e informáticos) y segundo por el enfoque diferente de las metodologías de investigación en ciencias de la computación y humanidades. Intenté escribir la tesis de una manera hacker, es decir, adoptar un enfoque creativo y alegre en el capítulo. No es una práctica tan común en la literatura comparada como lo es en la literatura de los programadores. La risa y la alegría son parte de sus preguntas. Geoffrey James lo dejó claro: si no fuera por la risa, no habría Tao. Además, como dijo, "incluso un programa perfecto todavía tiene errores". Mi trabajo, no es una excepción. Habrá errores que deberán ser corregidos o descubiertos a través de alguna otra “unidad” y “pruebas de integración” que no sean las mías.

Como dijo una vez Julian Jonker: “Hybrid experiences and immersive cultures don't trace ownership or home, but they do provide more room for the imagination to breathe.” [Jonker, 2013, p. 564]. Espero que la tesis pueda proporcionar algunos espacios para imaginar un poema que aún no ha sido escrito.

Bibliografía

- Aarseth, Espen J., *Hypertext, Cybertext, Digital Literature, Medium,* Dichtung Digital, Website, http://www.dichtung-digital.de/Interviews/Aarseth-16-Dez-99/, Accessed 4 Dec 2018, Germany, 1999.
- Aarseth, Espen J., *Cybertext: Perspectives on Ergodic Literature,* John Hopkins University Press, USA, 1997.
- Beiguelman, Giselle, "The Reader, the Player, and the Executable Poetics", *Beyond the Screen: Transformations of Literary Structures, Interfaces and Genre,* eds. Jörgen Schäffer and Peter Gendolla, Transcript Verlag, 2010, pp. 403-426.
- Benjamin, Walter, "The work of art in the age of mechanical reproduction", *The Work of Art in the Age of Its Technological Reproducibility, and Other Writings on Media,* Harvard University Press, USA, 2008.
- Berners-Lee, T., J. Hendler, and O. Lissila, 2001, "The Semantic Web¨, *Scientifc American,* USA, 2001.
- Block, Friedrich W., *p0es1s: The Aesthetics of Digital Poetry, eds. Christiane Heibach, and Karin Wenz,* Hatje Cantz Verlag, Germany, 2004.
- Bohn, Willard, *Modern Visual Poetry,* Associated University Presses, London, 2001.
- Bootz, Philippe, "Digital Poetry: From Cybertext to Programmed Forms", *New Media Poetry and Poetics, Special Issue, Leonardo Electronic Almanac:* Vol. 14, No. 5-6, 2016.

- Bootz, Philippe, ¨The Problematic of Form Transitoire Observable: A Laboratory For Emergent Programmed Art¨, *Dichtung-digital*, Germany, 2005
- Bootz, Philippe, "Towards an ontology of the field of digital poetry", *Electronic literature in Europe*, Université de Bergen, Norway, 2008.
- Brams, Steven, *Game Theory and the Humanities: Bridging Two Worlds*, The MIT Press, USA, 2011.
- Brandt, Per Aage, "Meaning and the Machine: Towards a Semiotics of Interaction", *The Computer as Medium*, eds. Peter B. Andersen, Berit Holmquist, and Jens F. Jensen, Cambridge University Press, England, 1993.
- Bresson, Robert, *Notes on Cinematography*, Urizen Books, USA, 1977.
- Burnett, Kathleen Burnett, ¨Towards a Theory of Hypertextual Design¨, *Postmodern Culture*, 3(2), pp. 1-12.
- Caillois, Roger, ¨The Definition of Play, The Classification of Games¨. *The Game Design Reader*, MIT Press, USA, 2006.
- Colapietro, Vincent M., Peirce's Approach to the Self: A Semiotic Perspective on Human Subjectivity, State University of New York Press, 1988, USA
- Crenshaw, Cheri, *Exploiting Kairos in Electronic Literature: A Rhetorical Analysis*, Texas Women's University, USA, 2008.
- Crawford, Chris, *The Art of Computer Game Design*, Washington State University, USA, 1997
- Csikszentmihalyi, Mihaly, ¨Implications of a systems perspective for the study of creativity¨, *Handbook of Creativity*, eds. R.J. Sternberg, Cambridge University Press, England, 1999.
- Darley, Andrew, *Visual Digital Culture*, Taylor & Francis, USA, 2000.
- Despa, Mihai Liviu, *Comparative study on software development methodologies*, Database Systems Journal, vol. V, no. 3, Economic Informatics and Cy-

bernetics Department, Bucharest University of Economic Studies, Romania, 2014, pp. 37-56

- Desrochers, Nadine & Tomaszek, Patricia, *Bridging The Unknown: An Interdisciplinary Case Study of Paratext in Electronic Literature, Examining Paratextual Theory and its Applications in Digital Culture*, Information Science Reference, 2014, USA
- Di Rosario, Giovanna, *Electronic Poetry: Understanding Poetry in the Digital Environment,* University of Jyvaskyla Press, Finland, 2011.
- Drucker, Joanna, *The Alphabetic Labyrinth: The Letters in History and Imagination,* Thames & Hudson, USA, 1999.
- Eskelinen, Markku, *Cybertext Poetics,* Bloomsbury Publishing, USA, 2012
- Lamb, Carolyn and others. "A Taxonomy of Generative Poetry Techniques". *Bridges Finland Conference*. Finland. 2016.
- Flores, Leonardo, *Typing the Dancing Signifier: Jim Andrews' (Vis)Poetics,* PhD dissertation, University of Maryland, USA, 2010.
- Funkhouser, C.T, *Prehistoric Digital Poetry: An Archaeology of Forms,* University of Alabama, USA, 2007.
- Gabbrielli, M. and S. Martini, *Programming Languages: Principles and Paradigms,* Springer, USA, 2010.
- Gilster, Paul, *Digital Literacy,* Wiley and Computer Publishing, USA, 1997.
- Gere, Charlie, *Digital Culture,* Reaktion Books, USA, 2009
- Glazier, Loss P., *Digital Poetics: The Making of E-Poetries,* University of Alabama Press, USA, 2001.
- Gruber, Thomas R, "A Translation Approach to Portable Ontology Specifications", *Knowledge Acquisition*, 5(2), Stanford University, USA, 1993. pp. 199-220.

- Hartman, Charles O., *Virtual Muse: Experiments in Computer Poetry,* The New England University Press, USA, 1996.
- Hayles, Katherine N., “Print is Flat, Code is Deep. The Importance of Media-Specific Analysis”, in *Poetics Today*, 25.1 2004, pp. 67-90.
- Hayles, Katherine N., ¨Translating Media: Why We Should Rethink Textuality?¨, *The Yale Journal of Criticism*, 16.2, 2003. pp. 263-290.
- Hayles, Katherine N., *Writing Machine,* The MIT Press, USA, 2002.
- Hayles, Katherine N., *Electronic Literature: New Horizons for the Literary,* University of Notre Dame Press, USA, 2008.
- Hayles, Katherine N., *Traumas of Code, Critical Digital Studies: A Reader*, University of Toronto Press, 2013, USA
- Hertz, David Michael, *The Tuning of the Word: The Musico-literary Poetics of the Symbolist Movement,* Southern Illinois University Press, USA, 1997.
- Jakobson, Roman, “Linguistics and Poetics”, *Style in Language,* eds. Thomas A. Sebeok, MIT Press, USA, 1960. pp. 350-377.
- Jonker, Julian. Black Secret Technology (The Whitey on the Moon Dub), Critical Digital Studies: A Reader, University of Toronto Press, 2013, USA
- Joyce, Michael, *Of Two Minds: Hypertext Pedagogy and Poetics*, University of Michigan Press, USA, 1996.
- Koskimaa, Raine, “Approaches to Digital Literature: Temporal Dynamics and Cyborg Authors”, *Reading Moving Letters,* eds. Roberto Simanowski, Jörgen Schäffer, and Peter Gendolla , Transcript Verlag, Germany, 2010. pp. 129-143.
- Kress, Gunther and Theo van Leeuwen, *Reading Images: The Grammar of Visual Design,* Routledge, England, 1996.

- Landow, George P., *Hypertext 2.0, the Convergence of Contemporary Critical Theory and Technology,* The John Hopkins University Press, USA, 1997.
- Landow, George P., *Hypertext 3.0: Critical Theory and New Media in an Era of Globalization (Parallax: Re-visions of Culture and Society),* The John Hopkins University Press, USA, 2006.
- Maciunas, George, *Fluxus: The History of an Attitude,* San Diego University Press, USA, 1998.
- Manovich, Lev, *The Language of New Media,* The MIT Press, USA, 2001.
- Marcus, Aaron, "Diagrammatic Visible Language: An Investigation of Visual Logic", *Leonardo*, Col. 20, No. 1, 1987, pp. 9-15.
- Martin, Robert C. Clean Code: A Handbook of Agile Software Craftsmanship, Pearson Education, Inc., 2009, USA
- McLuhan, Marshall, *Understanding Media: The Extensions of Man*, W. Terrence Gordon, USA, 1964
- Memmott, Talan, *Digital Rhetoric and Poetics: Signifying Strategies in Electronic Literature,* Malmo University, Sweden, 2011.
- Morales Benito, Lidia, "Las reglas del juego: teoría y práctica del juego en literatura", *Les Ateliers du SAL*, 1-2 (2012): 271-281.
- Morris, Adalaide and Thomas Swiss, *New Media Poetics: Contexts, Technotexts, and Theories,* The MIT Press, USA, 2006.
- Manurung, H, *An evolutionary algorithm approach to poetry generation*. PhD thesis, University of Edinburgh, 2004.
- Nelson, Jason, ¨Digital Poetry: A Revolution In Publishing?¨, Huffpost, May 2011. https://www.huffingtonpost.com/jason-nelson/digital-poetry_b_824768.html.

- Mark, Overmars, *"A brief history of computer games"*, University of Utrech, Germany, 2012. https://issuu.com/2epal/docs/history_of_games.
- Murray, Janet, *Hamlet on the Holodeck: The Future of Narrative in Cyberspace*, MIT Press, USA, 1998.
- Peirce, C. S. (1931-1958). Collected Papers (Vols. 1-8). Cambridge: Harvard University Press, USA,
- Ray, Fernando Gonzalez, Advances in Subjectivity from a Cultural-Historical Perspective: Unfoldings and Consequences for Cultural Studies Today, Perezhivanie, Emotions and Subjectivity: Advancing Vygotsky's Legacy, Springer. 2017, USA
- Ricardo, F. J., *Literary Art in Digital Performance: Case Studies in New Media Art and Criticism*, Bloomsbury Publishing, USA, 2009.
- Sawyer, R. Keith, *Explaining Creativity: The Science of Human Innovation*, Oxford University Press, England, 2006.
- Srinivasan, Venkatesh, Synthesis of Machine Code: Algorithms and Applications, PhD thesis, University of Wisconsin, 2017, USA.
- Tabbi, Joseph, ¨Toward a Semantic Literary Web: Setting a Direction for the Electronic Literature Organization's Directory¨, *Poetics Today*. 38 (2010). pp. 17-50.
- Tanenbaum, Andrew S., *Modern Operating Systems*, Prentice Hall, USA, 2014
- Srinivasan, Venkatesh, Synthesis of Machine Code: Algorithms and Applications, PhD thesis, University of Wisconsin, 2017, USA.
- Van Dijk, Yra. Amateurs online: Creativity in a community, Electronic Literature Communities, Center for Literary Computing and ELMCIP, 2015, USA

- Vatrapu, Ravi & Suthers, Dan, Culture and Computers: A Review of the Concept of Culture and Implications for Intercultural Collaborative Online Learning, University of Hawaii at Manoa, 2007, USA
- Wardrip-Fruin, N., "Learning to Read Digital Literature", *Reading Moving Letters: Digital Literature in Research and Teaching: A Handbook*, Transcript Verlag, Germany, 2015. pp. 249-259.
- Weisberg, Robert W., *Creativity: Understanding Innovation in Problem Solving, Science, Invention, and the Arts,* Wiley, USA, 2006.
- Welty, Ch. and B. Smith, *Formal Ontology in Information Systems,* ACM Press, USA, 2011.
- Whicker, J. H., *Object-Oriented Writing Theory: Writers, Texts, Ecologies,* Ohio University, USA, 2014.
- Whitehead, Alfred North, ¨Process and Reality: an essay in cosmology¨, *Gifford Lectures delivered in the University of Edinburgh during the session 1927-28*, Free Press, USA, 1979.
- Wittgenstein, Ludwig, *Philosophical Investigations*, trans. G. E. M. Anscombe, Basil Blackwell, England, 1972.
- W3C, "The Semantic Web Activity", *The World Wide Web Consortium*, 2001. https://www.w3.org/2001/sw/
- Zambrano, Maria, *Filosofía y poesía*, Fondo de Cultura Económica, México, 2012.
- 2Bears, Jackson, A Conversation with Spirits inside the Simulation of a Coast Salish Longhouse, Critical Digital Studies: A Reader, University of Toronto Press, 2013, USA